Optimasi *Multiple Travelling Salesmen Problem* Distribusi Produk PT Indomarco Adi Prima (Stock Point Nganjuk) dengan menggunakan Metode K-Means dan Algoritma Genetika (GKA)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Yusuf Afandi
NIM: 175150218113049

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2021

PENGESAHAN

Optimasi *Multiple Travelling Salesmen Problem* Distribusi Produk PT Indomarco Adi Prima (*Stock Point* Nganjuk) dengan menggunakan Algoritma *K-Means* dan Algoritma Genetika (GKA)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Yusuf Afandi
NIM: 175150218113049

Skripsi ini telah diuji dan dinyatakan lulus pada
20 Juli 2021
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Imam Cholissodin, S.Si., M.Kom.	Bayu Rahayudi, M.T., M.M.
NIK: 201201 850719 1 001	NIP: 19740712 200604 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Achmad Basuki, S.T., M.MG., Ph.D.
NIP: 19741118 200312 1 002

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 23 Juli 2021

Yusuf Afandi

NIM: 175150218113049

PRAKATA

Syukur Alhamdulillah penulis panjatkan kehadirat Allah SWT atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan penelitian yang berjudul "Optimasi *Multiple Travelling Salesmen Problem* Distribusi Produk PT Indomarco Adi Prima (*Stock Point* Nganjuk) dengan menggunakan Algoritma *K-Means* dan Algoritma Genetika (GKA)" ini dengan baik.

Penulis menyadari bahwa penelitian ini tidak akan terselesaikan tanpa adanya bantuan, doa, serta kemurahan hati dari berbagai pihak. Oleh karena itu, disamping rasa syukur yang tak terhingga atas pencapaian ini, penulis ingin mengucapkan rasa terima kasih yang sebesar-besarnya kepada:

1. Bapak Imam Cholissodin, S.Si., M.Kom., selaku Dosen Pembimbing I yang telah membimbing penulis dengan sabar dan teliti dalam menyelesaikan penelitian ini.
2. Bapak Bayu Rahayudi, M.T, M.M., selaku Dosen Pembimbing II yang dengan ikhlas memberikan arahan dan bimbingan penulis dalam penyusunan penelitian ini.
3. Bapak Adhitya Bhawiyuga, S.Kom., M.Sc., selaku Ketua Program Studi Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Achmad Basuki, S.T., M.MG., Ph.D., selaku Ketua Jurusan Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
5. Keluarga penulis, terutama kedua orang tua dan adik yang selalu memberikan doa, dukungan, motivasi, dan kesabaran dalam segala hal.
6. Seluruh pihak yang belum dicantumkan namanya yang telah mendukung dan membantu penulis dalam menyelesaikan penelitian ini dengan tulus dan ikhlas.

Penulis juga menyadari bahwa pada penelitian ini masih memiliki kekurangan, oleh karena itu penulis sangat mengharapkan kritik dan saran yang membangun. Semoga penelitian ini dapat berguna dan membantu penelitian selanjutnya.

Malang, 20 Juli 2021

Yusuf Afandi

Afandyy97@gmail.com

ABSTRAK

Yusuf Afandi, Optimasi *Multiple Travelling Salesmen Problem* Distribusi Produk PT Indomarco Adi Prima (*Stock Point* Nganjuk) dengan menggunakan Algoritma K-Means dan Algoritma Genetika (GKA)

Pembimbing: Imam Cholissodin, S.Si., M.Kom. dan Bayu Rahayudi, M.T., M.M.

Distribusi merupakan salah satu hal yang sangat penting agar suatu produk bisa tersampaikan kepada pelanggan/konsumen. Tugas seorang *salesman* adalah mengunjungi toko atau pelanggan satu hari sebelum barang dikirim untuk menawarkan produk dan mencatat barang pesanan. Faktor yang harus diperhatikan dalam proses tersebut adalah waktu dan biaya yang dibutuhkan harus seminimal mungkin. Salah satu aspek yang dapat memengaruhi kedua faktor tersebut tersebut adalah rute perjalanan yang harus optimal. Pada penelitian kali ini permasalahan yang akan coba diselesaikan adalah tentang optimasi rute distribusi produk pada PT indomarco Adi Prima (Stock Point Nganjuk) yang memiliki beberapa titik untuk dikunjungi oleh lebih dari satu *salesman* satu hari sebelum pendistribusian barang. Terdapat beberapa proses untuk menyelesaikan permasalahan tersebut, yaitu input data berupa data *latitude* dan *longitude*, kemudian data tersebut akan dibagi menjadi beberapa klaster sesuai dengan jumlah *salesman* pada perusahaan, selanjutnya setiap klaster tersebut akan dicari rute terpendeknya untuk 5 hari kerja dan hasil akhirnya seluruh rute terpendek dari setiap klaster akan dihitung total jarak tempuhnya. Berdasarkan dari hasil pengujian yang telah dilakukan menggunakan data *latitude* dan *longitude* yang diperoleh dari PT indomarco Adi Prima (Stock Point Nganjuk) diperoleh hasil rute yang paling optimal dengan total jarak tempuh 259,722337 km dengan *fitness* 0,385026. Parameter yang paling optimal adalah ketika ukuran populasi 1600, jumlah generasi 800 dengan kombinasi nilai *crossover rate* (*cr*) 0,4 dan *mutation rate* (*mr*) 0,6.

Kata kunci: optimasi, rute distribusi, *K-Means*, Algoritma Genetika

ABSTRACT

Yusuf Afandi, Optimization of *Multiple Travelling Salesmen PT Indomarco Adi Prima's Product Distribution (Stock Point Nganjuk) using the K-Means algorithm and Genetic Algorithm (GKA)*

***Supervisors:* Imam Cholissodin, S.Si., M.Kom. *and* Bayu Rahayudi, M.T., M.M.**

Distribution is one thing that is very important so that a product can be delivered to customers / consumers. The job of a salesman is to visit the store or customer one day before the goods are shipped to offer the product and record the ordered goods. Factors that must be considered in the process are the time and costs required to be as minimal as possible. One aspect that can influence these two factors is the optimal travel route. In this study, the problem that we will try to solve is the optimization of product distribution routes at PT Indomarco Adi Prima (Stock Point Nganjuk) which has several points to be visited by more than one salesman one day before the distribution of goods. There are several processes to solve these problems, namely input data in the form of latitude and longitude data, then the data will be divided into several clusters according to the number of salesmen in the company, then each cluster will look for the shortest route for 5 working days and the end result is all the shortest routes from each cluster the total distance traveled will be calculated. Based on the results of tests that have been carried out using latitude and longitude data obtained from PT Indomarco Adi Prima (Stock Point Nganjuk), the most optimal route results with a total distance of 259.722337 km with a fitness of 0.385026. The most optimal parameter is when the population size is 1600, the number of generations is 800 with a combination of crossover rate (cr) 0.4 and mutation rate (mr) 0.6.

Keywords: optimization, distribution routes, K-Means, genetic algorithm

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Distribusi atau pelaksanaan penempatan produk memiliki tujuan untuk mendekatkan produk kepada calon konsumen atau target pasarnya. Perusahaan distribusi membantu perusahaan untuk menyimpan dan mendistribusikan barang ke tujuan. Tugas seorang *salesman* adalah harus mengunjungi titik kunjungan satu hari sebelum barang didistribusikan untuk menawarkan produk dan mencatat barang yang dipesan toko atau pelanggan. Perusahaan harus menentukan cara terbaik dalam proses kunjungan dengan memperhatikan faktor biaya dan waktu yang dibutuhkan (2005). Salah satu aspek yang memengaruhi kedua faktor tersebut adalah rute perjalanan *salesman* yang harus optimal, sehingga biaya dan waktu yang dibutuhkan bisa diminimalkan.

Pada penelitian kali ini permasalahan yang akan coba diselesaikan adalah tentang optimasi rute pendistribusian produk pada PT Indomarco Adi Prima (*Stock Point Nganjuk*) yang memiliki cukup banyak titik yang harus dikunjungi. Pada kasus normal permasalahan tersebut masuk ke dalam kategori *Travelling Salesman Problem* (TSP), dengan seorang *salesman* harus bisa mengunjungi semua titik tepat satu kali dan harus kembali ke titik awal dengan jarak seminimal mungkin. Namun permasalahan yang akan diselesaikan kali ini berbeda, salah satu admin gudang di perusahaan tersebut mengatakan bahwa terdapat tiga *salesman* yang beroperasi untuk mengunjungi toko atau pelanggan dan seringkali *salesman* yang bertugas hanya mengandalkan daftar toko yang akan dikunjungi tanpa memperhatikan biaya dan waktu yang dibutuhkan. Sehingga permasalahan pada studi kasus ini termasuk ke dalam kategori *Multiple Traveling Salesman Problem* (MTSP).

Untuk menyelesaikan studi kasus MTSP, salah satu algoritma yang sering digunakan adalah Algoritma Genetika. Algoritma ini dipilih karena menggunakan teknik optimasi berbasis populasi yang mampu menyediakan ragam solusi terbaik pada sebuah populasi di setiap generasi berdasarkan nilai *fitness* tertinggi. Proses ini dilakukan berkali-kali sehingga dapat mensimulasikan proses evolusi yang semakin baik. Hasil akhir yang diperoleh direpresentasikan sebagai kromosom yang mempunyai nilai *fitness* tertinggi dari seluruh generasi. Namun, untuk permasalahan MTSP ini ketika data berjumlah banyak dan data tidak terstruktur dengan baik berdasarkan jarak yang berdekatan, Algoritma Genetika akan mengalami kesulitan dan membutuhkan waktu komputasi yang lama untuk memperoleh hasil yang optimal (2016). Untuk mengatasi hal tersebut, algoritma yang akan digunakan pada penelitian kali ini adalah K-Means dan Algoritma Genetika. K-Means digunakan untuk melakukan pengelompokan titik-titik yang akan dikunjungi sehingga akan menghasilkan kelompok titik-titik yang jaraknya berdekatan. Kemudian Algoritma Genetika digunakan untuk mencari rute terpendek dari setiap kelompok titik-titik yang terbentuk untuk 5 hari kerja. Sebelumnya penelitian-penelitian yang menggunakan metode gabungan K-Means

dan Algoritma Genetika sudah pernah dilakukan, salah satunya adalah penelitian yang dilakukan oleh Zhanqing Lu, Kai Zhang, Juanjuan He, dan Yunyun Niu. Dalam penelitiannya, mereka menggunakan K-Means untuk melakukan pengelompokan terhadap semua titik berdasarkan jarak terdekat sekaligus mencari *starting point* untuk Algoritma Genetika mencari rute paling optimal. Hasil penelitian menunjukkan bahwa kombinasi antara K-Means dengan Algoritma Genetika lebih baik daripada hanya menggunakan Algoritma Genetika tradisional (Zhanqing Lu, Kai Zhang, Juanjuan He 2016).

Pada penelitian lain yang dilakukan oleh Adyan Nur Alfiyatin, Wayan Firdaus Mahmudy, Yusuf Priyo Anggodo, kombinasi dari K-Means dan Algoritma Genetika juga memiliki hasil yang lebih baik dibandingkan dengan Algoritma Genetika tradisional. Hal tersebut terlihat bahwa hasil yang baik juga tergantung pada kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr)* sehingga menghasilkan *fitness* yang baik (Alfiyatin, Mahmudy, and Anggodo 2018).

Berdasarkan beberapa penelitian di atas, maka pada penelitian kali ini dengan topik "Optimasi *Multiple Travelling Salesmen Problem* Distribusi Produk PT Indomarco Adi Prima (Stock Point Nganjuk)" akan menggunakan metode gabungan K-Means dan Algoritma Genetika (GKA). Dengan memilih metode tersebut diharapkan dapat memberikan hasil yang terbaik dan bisa digunakan sebagai bahan pertimbangan bagi perusahaan dalam menentukan rute yang optimal untuk rencana perjalanan *salesman*.

1.2 Rumusan Masalah

1. Bagaimana nilai parameter yang terbaik K-Means dan Algoritma Genetika untuk menyelesaikan permasalahan dalam penelitian ini?
2. Bagaimana tingkat perbandingan hasil antara K-Means dan Algoritma Genetika dengan Algoritma Genetika tanpa K-Means?
3. Bagaimana perbandingan jarak tempuh yang dihasilkan K-Means dan Algoritma Genetika dengan rute yang digunakan perusahaan saat ini?

1.3 Tujuan

1. Mengetahui nilai parameter terbaik pada Algoritma K-means dan Algoritma Genetika untuk menyelesaikan permasalahan dalam penelitian ini.
2. Membandingkan hasil antara K-Means dan Algoritma Genetika dengan Algoritma Genetika tanpa K-Means.
3. Membandingkan hasil antara K-Means dan Algoritma Genetika dengan rute yang digunakan perusahaan saat ini.

1.4 Manfaat

1. Diharapkan hasil dari penelitian ini bisa menjadi bahan pertimbangan bagi perusahaan atau pihak terkait dalam menentukan rute rencana perjalanan *salesman* yang efektif dan efisien.

2. Dapat dijadikan sebagai landasan penelitian selanjutnya untuk permasalahan yang sejenis.

1.5 Batasan Masalah

1. Proses kunjungan hanya dilakukan dari Gudang PT Indomarco Adi Prima Nganjuk ke pelanggan (toko).
2. Jarak tempuh setiap titik kunjungan *sales* dihitung mulai dari titik awal keberangkatan, yaitu PT Indomarco Adi Prima (*Stock Point Nganjuk*) dan berakhir juga PT Indomarco Adi Prima (*Stock Point Nganjuk*).
3. Jumlah *sales* yang beroperasi setiap harinya adalah sama.
4. Kunjungan hanya dilakukan tepat satu kali pada setiap titik dalam sekali putaran sehingga tidak ada pengulangan ke titik yang sudah dikunjungi.
5. Kondisi jalan seperti macet, selama proses distribusi tidak dijadikan sebagai parameter.

1.6 Sistematika Pembahasan

Sistematika pembahasan pada penelitian kali ini adalah sebagai berikut:

BAB 1: PENDAHULUAN

Pada bab ini akan dijelaskan mengenai latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, serta sistematika pembahasan.

BAB 2: LANDASAN KEPUSTAKAAN

Pada bab ini akan dijelaskan mengenai beberapa teori pendukung serta penelitian-penelitan telah dilakukan sebelumnya demi menunjang pengerjaan penelitian ini, yaitu yang berkaitan dengan metode klasterisasi dan optimasi. Selain itu pada bab ini juga akan dijelaskan tentang beberapa hal yang berkaitan dengan penelitian ini. Referensi yang digunakan berasal dari berbagai sumber, seperti jurnal, buku, dan *International conference* dan beberapa sumber lainnya.

BAB 3: METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan tentang langkah-langkah secara sistematis mengenai pengerjaan sistem yang akan diimplementasikan untuk menyelesaikan permasalahan Optimasi *Multiple Travelling Salesmen Problem* Distribusi Produk PT Indomarco Adi Prima (Stock Point Nganjuk) dengan menggunakan metode K-Means dan Algoritma Genetika (GKA) dengan data berupa jarak yang diperoleh dari titik koordinat toko pelanggan yang dikunjungi *salesman* untuk distribusi produk di wilayah Nganjuk.

BAB 4: PERANCANGAN

Pada bab ini akan dijelaskan tentang perancangan yang dilakukan dalam membuat sistem untuk menyelesaikan permaslahan Optimasi *Multiple Travelling Salesmen Problem* Distribusi Produk PT

Indomarco Adi Prima (Stock Point Nganjuk) dengan menggunakan metode K-Means dan Algoritma Genetika (GKA) dengan data berupa jarak yang diperoleh dari titik koordinat toko pelanggan yang dikunjungi *sales* untuk distribusi produk di wilayah Nganjuk.

BAB 5: IMPLEMENTASI

Pada bab ini akan dijelaskan tentang teknik implementasi, batasan-batasan dalam melakukan implementasi dan metode operasi yang digunakan dalam pengembangan sistem.

BAB 6: PENGUJIAN DAN ANALISIS

Pada bab ini akan dijelaskan tentang proses serta hasil pengujian metode yang digunakan pada sistem dan sesuai dengan perancangan sistem dan dilengkapi dengan analisis dari hasil yang keluar terhadap metode yang digunakan.

BAB 7: PENUTUP

Bab ini berisi kesimpulan dari hasil penelitian beserta saran atas kekurangan pada penelitian, sehingga dapat dilakukan pengembangan menjadi penelitian yang lebih baik lagi ke depannya.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini akan dijelaskan mengenai beberapa teori pendukung serta penelitian-penelitan terkait yang telah dilakukan sebelumnya demi menunjang pengerjaan penelitian ini, yaitu yang berkaitan dengan metode klasterisasi dan optimasi. Selain itu pada bab ini juga akan dijelaskan tentang PT Indomarco Adi Prima, serta tentang apa itu distribusi, optimasi, penjelasan tentang metode yang digunakan, yaitu K-Means dan Algoritma Genetika.

2.1 Isi Landasan Kepustakaan

Kajian pustaka berisi uraian tentang penelitian yang sebelumnya sudah pernah dilakukan dengan menggunakan metode yang sama dengan penelitian ini. Bahan kajian Pustaka yang digunakan merupakan penelitian yang membahas mengenai teori beserta tinjauan pustaka yang berkaitan dengan kombinasi metode K-Means dan Algoritma Genetika.

Penelitian yang menggabungkan K-Means dan Algoritma Genetika sebelumnya sudah pernah dilakukan oleh Zhanqing Lu, Kai Zhang, Juanjuan He, dan Yunyun Niu. Penelitian tersebut bertujuan untuk mencari rute paling optimum untuk kendaraan udara tak berawak atau *drone* untuk keperluan militer dan sipil dengan beberapa *drone* yang difungsikan. Dalam penelitiannya, mereka menggunakan K-Means untuk melakukan *clustering* terhadap semua titik berdasarkan jarak terdekat sekaligus mencari *starting point* untuk Algoritma Genetika mencari rute paling optimal. Hasil penelitian menunjukkan bahwa kombinasi metode K-Means dengan Algoritma Genetika lebih baik daripada hanya menggunakan Algoritma Genetika tradisional (2016). Penelitian yang sama juga pernah dilakukan oleh Amirah Rahman, Wai Loan Liew, Tan, dan Shahruddin yang bertujuan untuk mengoptimalkan rute pengiriman surat oleh beberapa agen. Dalam penelitiannya, mereka menguraikan permasalahan MTSP menjadi beberapa TSP terpisah dengan mengelompokkan titik-titik kunjungan dengan menggunakan K-Means, selanjutnya menyelesaikan setiap klaster TSP menggunakan Algoritma Gentika. Hasilnya menunjukkan bahwa metode tersebut mampu menyediakan rute yang lebih baik untuk sistem pengiriman surat dibandingkan dengan rute asli, dengan selisih jarak 11,25% (Rahman et al., 2014).

Penelitian selanjutnya masih terkait dengan metode K-Means yang dikombinasikan dengan Algoritma Genetika. Penelitian ini dilakukan oleh Adyan Nur Alfiyatin, Wayan Firdaus Mahmudy, Yusuf Priyo Anggodo (2018). Penelitian ini bertujuan untuk menyelesaikan masalah perutean kendaraan dengan *Vehicle Routing Problem with Time Windows* (VRPTW). VRP merupakan permasalahan kombinatorial yang membahas tentang proses pendistribusian barang dari depo (pusat distribusi) ke pelanggan yang tersebar di berbagai lokasi dengan kendaraan terbatas, jarak depo dan pelanggan, kapasitas kendaraan dan waktu. VRP dengan *Time Window* (VRPTW) terdiri dari sekumpulan pelanggan yang dilayani oleh satu kendaraan dari depot utama dan masing-masing kendaraan. Metode yang digunakan adalah gabungan K-Means dan Algoritma Genetika. K-means dapat

melakukan *clustering* dengan baik dan Algoritma Genetika dapat mengoptimalkan rute tersebut. Algoritma Genetika yang diusulkan menggunakan inisialisasi kromosom dari hasil *K-Means* dan menggunakan metode pemilihan pengganti. Berdasarkan perbandingan antara Algoritma Genetika dan Algoritma Genetika *hybrid* K-Means membuktikan bahwa Algoritma Genetika *hybrid* K-Means merupakan kombinasi yang cocok dengan waktu komputasi yang relatif rendah, yaitu perbandingan antara 2700 dan 3900 detik (Alfiyatin, Mahmudy and Anggodo, 2018).

Berdasarkan ketiga penelitian sebelumnya dapat disimpulkan bahwa untuk mencari rute paling optimum dari permasalahan rute rencana perjalanan *salesman* dengan kasus *Multiple Travelling Salesman Problem* (MTSP) dapat dilakukan dengan menggunakan kombinasi metode K-Means dan Algoritma Genetika dengan hasil yang lebih baik dibandingkan dengan Algoritma Genetika saja. Hal tersebut membuat penulis semakin yakin untuk menggunakan gabungan kedua metode tersebut untuk menyelesaikan permasalahan rute rencana perjalanan *salesman* di PT Indomarco Adi Prima (Stock Point Nganjuk).

2.2 Optimasi Rute Distribusi Barang PT Indomarco Adi Prima

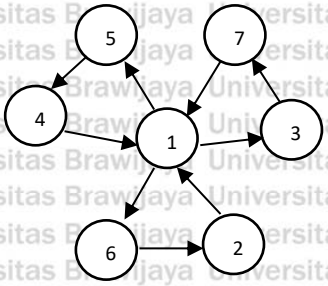
Distribusi merupakan proses pelaksanaan penempatan produk memiliki tujuan untuk mendekatkan produk kepada calon konsumen atau target pasarnya (2015). Tugas dari *salesman* adalah harus mengunjungi seluruh titik toko atau pelanggan guna menawarkan produk dan mencatat barang yang dipesan toko atau pelanggan tersebut tepat satu hari sebelum barang pesanan didistribusikan. Dalam proses kunjungan, faktor yang harus diperhatikan adalah efektifitas waktu dan efisiensi biaya, jika keduanya tidak terpenuhi, maka akan berpengaruh pada tingkat kepuasan konsumen dan biaya operasional yang dikelauarkan oleh perusahaan. Salah satu hal yang bisa dilakukan untuk memenuhi hal tersebut adalah melakukan optimasi rute rencana perjalanan *salesman*. Dengan melakukan optimasi rute akan memberikan keuntungan pada penyedia layanan, pelanggan, dan lingkungan karena jarak tempuh yang pendek berarti biaya operasional lebih rendah serta kepuasan pelanggan yang lebih besar (Rahman et al., 2014).

PT Indomarco Adi Prima merupakan jaringan distribusi PT Indofood yang juga bergerak di bidang logistik sebagai *3rd Party Logistics* (3PL) bagi grup maupun non grup. PT Indomarco Adi Prima memiliki banyak sekali cabang, salah satunya yang ada di Nganjuk. Terdapat 3 *salesman* yang biasanya beroperasi untuk mengunjungi banyak toko atau minimarket yang berada di area Nganjuk. Pada saat proses kunjungan, setiap *salesman* yang beroperasi hanya mengandalkan daftar toko pelanggan yang akan dikunjungi tanpa memperhatikan efektivitas jarak yang akan ditempuh. Produk yang didistribusikan biasanya adalah produk kebutuhan sehari-hari, terutama produk dari Indofood.

2.3 *Multiple Travelling Salesman Problem* (MTSP)

Multiple Travelling Salesman Problem (MTSP) merupakan permasalahan yang mirip dengan *Travelling Salesman Problem* (TSP). Bedanya adalah jika TSP

hanya terdapat seorang *sales* dan harus bisa mengunjungi semua titik tepat satu kali dan harus Kembali ke titik awal dengan jarak seminimal mungkin. Sedangkan MTSP memilliki *m* jumlah *sales* yang harus mengunjungi *n* jumlah kota dalam sekali waktu dengan jarak seminimal mungkin (Rostami et al., 2015).



Gambar 2.1 Ilustrasi MTSP
Sumber : (Rostami et al., 2015)

Seperti yang terlihat pada Gambar 2.2 Ilustrasi MTSP, node 1 merupakan titik awal dan akhir distribusi. Dalam gambar tersebut juga terlihat bahwa terdapat beberapa klaster node yang merepresentasikan toko yang akan dikunjungi *sales*. Banyak klaster disesuaikan dengan jumlah *sales* yang ada, misal pada gambar di atas jumlah *sales* ada 3, maka klaster yang digunakan sebanyak tiga klaster. Setiap klaster akan dikunjungi oleh satu *sales* tepat satu kali, misal klaster 1 yaitu node 4 dan 5 akan dikunjungi oleh *sales* A dengan urutan rute 1→5→4→1. Sedangkan klaster 2 yaitu node 3 dan 7 akan dikunjungi oleh *sales* B dengan urutan rute 1→3→7→1. Dan terakhir adalah klaster 3 yaitu node 2 dan 6 akan dikunjungi *sales* C dengan urutan rute 1→2→6→1.

2.4 K-Means

K-Means merupakan algoritma yang membagi data ke dalam klaster-klaster berdasarkan kemiripannya dan setiap klaster memiliki perbedaan. secara lebih detail menurut Sarwono (2010), K-means merupakan metode yang mempartisi atau membagi data ke dalam beberapa kelompok berdasarkan kemiripan karakteristiknya. Sehingga data yang mempunyai tingkat *similarity* yang tinggi karakteristiknya akan dikelompokkan ke dalam klaster yang sama dan data yang tingkat *similarity* rendah akan dikelompokkan ke dalam kelompok yang lainnya. K-Means menurut Sarwono (2010) adalah sebagai berikut:

1. Menentukan nilai *k* sebagai nilai banyaknya klaster yang ingin dibuat.
2. Membangkitkan nilai *centroid* atau titik pusat klaster awal secara acak sebanyak *k*.
3. Menghitung jarak setiap data masukan terhadap setiap *centroid* menggunakan rumus jarak *euclidian* pada Persamaan 2.1

$$\mathbf{d}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \qquad (2.1)$$

d = perhitungan jarak dari titik menuju *centroid*

n = banyak data yang akan dihitung jaraknya

i=1 = proses klasterisasi dumulai dari 1

x_i = titik koordinat objek ke-*i*

y_i = titik koordinat centroid ke-*i*

4. Mengelompokkan data sesuai jarak (sesuai hasil perhitungan Langkah 3) yang berdekatan dengan *centroid* (jarak terkecil).
5. Memperbaharui nilai *centroid* dengan Persamaan 2.2

$$\boldsymbol{\mu_j = \frac{1}{N_j} \sum_{j \in S_j} x_i} \quad (2.2)$$

μ_j = titik *centroid* baru pada klaster *j*

N_j = banyaknya data pada klaster *j*

x_i = data ke-i pada klaster *j*

6. melakukan perulangan mulai dari langkah ke 2 hingga langkah ke 5.
7. jika sudah terpenuhi maka iterasi terakhir akan digunakan sebagai parameter untuk klasifikasi data.

K-Means merupakan algoritma yang sangat popular karena banyak digunakan dalam pengelompokan data. Algoritma ini sering digunakan karana kesederhanaan dalam implementasinya, efisiensi dan kemudahan konvergensi. Namun algoritma K-Means ini memiliki kelemahan, yaitu pada saat inisialisasi *centroid* awal yang dilakukan secara acak, jika nilai acak yang dihasilkan kurang baik, maka hasil pengelompokan data bisa berbeda-beda dan hasil pengelompokannya bisa jadi kurang optimal. Menurut Kumar and Sahoo (2014), hal tersebut dapat diatasi dengan menggunakan pendekatan *binary search algorithm* pada saat inisialisasi *centroid* awal. Berikut adalah algoritmanya:

1. Inisialisasi jumlah klaster (*k*), nilai *k* =1,2,3…m.
2. Menghitung jarak spesifik antar *centroid* yang akan dibuat, dengan Persamaan 2.3

$$M = \frac{max(A_i) - min(A_i)}{k} \quad (2.3)$$

3. Menghitung nilai centroid awal dengan Persamaan 2.4

$$C_k = \min(A_i) + (k-1)M \quad (2.4)$$

4. Langkah berikutnya sama dengan K-Means pada umumnya.

2.5 Haversine

Haversine adalah sebuah metode yang digunakan untuk menghitung jarak dari dua titik pada permukaan bumi dengan menggunakan garis lintang (*latitude*) dan garis bujur (*longitude*) (2013). Formula ini sangat penting dalam sistem navigasi, karena akan menghasilkan jarak terpendek dari dua titik. Rumus haversine dapat dilihat pada Persamaan 2.5

$$\boldsymbol{d = 2r.\arcsin\left\{\sqrt{sin^2\left(\frac{Lat_1 - Lat_2}{2}\right) + cos(Lat_1).\cos(Lat_2).sin^2\left(\frac{Long_1 - Long_2}{2}\right)}\right\}} \quad (2.5)$$

d = jarak

r = radius bumi (6371 untuk satuan km)

Lat_1 = *Latitude* titik 1

Lat_2 = *Latitude* titik 2

$Long_1$ = *Longitude* titik 1

$Long_2$ = *Longitude* titik 2

2.6 Algoritma Genetika

Algoritma Genetika adalah pendekatan komputasional untuk penyelesaian masalah yang dimodelkan seperti proses evolusi biologis. Algoritma ini terinspirasi oleh teori evolusi Darwin, karena mekanisme utama yang disimulasikan dalam algoritma ini adalah *survival of the fittes,* yaitu organisme yang kuat kemungkinan bertahan hidupnya lebih tinggi. Algoritma Genetika menggunakan tiga operator utama untuk meningkatkan kromosom di setiap generasi, yaitu : seleksi, persilangan (rekombinasi), dan mutasi (Mirjalili et al., 2020).

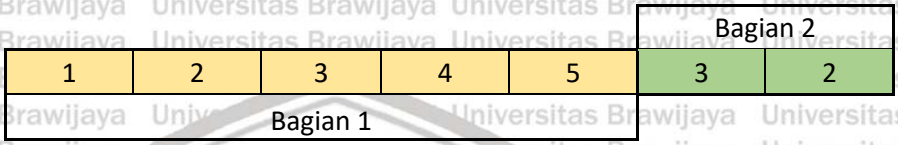
Menurut Sayyidah Karimah, Agus Wahyu Widodo, dan Imam Cholissodin (2017), struktur Algoritma Genetika adalah sebagai berikut:

1. Membangkitkan populasi awal
   Pada tahap ini yang dilakukan adalah melakulan inisialisasi populasi awal secara acak yang sekaligus menjadi solusi awal. Populasi ini terdiri atas sejumlah kromosom yang merepresentasikan solusi yang diinginkan.
2. Proses reproduksi
   Tahap ini dilukan untuk mendapatkan sebuah keturunan melalui operator genetik *crossover* dan mutasi. *Crossover* merupakan proses persilangan gen atara dua individu (induk) yang menghasilkan dua individu baru(*offspring*) pada generasi selanjutnya. Sedangkan mutasi merupakan proses mengubah nilai gen dalam satu kromosom. Mutasi menghasilkan individu baru dengan melakukan modofikasi terhadap satu gen atau lebih dalam individu yang sama.
3. Proses Evaluasi
   Pada Tahap ini seluruh *child* hasil dari proses reproduksi akan digabungkan menjadi satu dengan *parent*, kemudian dihitung nilai *fitness* pada setiap individu. Rumus perhitungan *fitness* bisa dilihat pada persamaan 2.6
4. Seleksi
   Pada tahap ini akan dilakukan seleksi terhadap seluruh individu berdasarkan nilai *fitness* yang terlah dihitung sehingga diperoleh solusi terbaik.

Dalam proses reproduksi terdapat banyak sekali jenis metode *crossover* dan mutasi yang bisa digunakan, namun ada beberapa yang biasa digunakan untuk menyelesaikan permasalahan TSP salah satunya adalah *Order Crossover* (OX) untuk *crossover* dan *Simple Inversion Mutation* untuk mutasi.

2.6.2 Representasi Kromosom

Langkah awal yang dilakukan dalam proses Algoritma Genetika adalah melakukan inisialisasi himpunan solusi baru yang tersusun atas sejumlah string kromosom yang berada dalam satu set populasi yang dibangkitkan secara acak. Representasi kromosom sangat penting untuk dilakukan karena digunakan untuk menjelaskan setiap individu dalam suatu populasi. Terdapat berbagai macam representasi pada kasus MTSP, salah satunya adalah kromosom yang terdiri dari dua bagian (2016). Untuk lebih jelas dapat dilihat pada Gambar 2.2.



Gambar 2.2 Representasi Kromosom

Pada bagian 1 menunjukkan urutan rute yang harus dilalui oleh sales, sedangkan bagian 2 menunjukkan banyak titik yang harus dikunjungi satu sales setiap harinya. Jadi pada Gambar 2.2, untuk hari pertama terdapat 3 titik yang harus dikunjungi, yaitu 1, 2, dan 2. Sedangkan pada hari kedua terdapat 2 titik yang harus dikunjungi, yaitu 4 dan 5.

2.6.3 Order Crossover (OX)

Order Crossover (OX) merupakan salah satu metode yang digunakan untuk melakukan *crossover* pada proses reproduksi Algoritma Genetika. OX pertama kali diusulkan oleh Lawrence Davis pada tahun 1991. Metode yang sering digunakan dalam Algoritma Genetika untuk menyelesaikan permasalahan TSP ini memiliki 2 variasi (Umbarkar dan Sheth 2015).

1. Variasi pertama *Order Crossover*

   Offspring akan terbentuk dengan memilih 2 individu yang akan menjadi induk. Misalnya adalah (2 4 3 5 6 8 9 1) dan (7 8 2 3 4 5 6 1) dan titik potongnya 2 dan 5, sehingga akan terbentuk (2 4 | 3 5 6 | 8 9 1) dan (7 8 | 2 3 4 | 5 6 1). Setelah itu salin elemen yang berada di antara titik potong untuk menjadi *offspring*, sehingga bentuknya menjadi (* * | 3 5 6 | * * *) dan (* * | 2 3 4 | * * *). Selanjutnya mulai dari titik potong kedua pada masing-masing induk pada elemen yang tidak ada di masing-masing *offspring* akan disalin ke *offspring* secara bersilangan. Ketika *offspring* bagian belakang sudah terisi penuh, maka langah menyalin dilanjutkan mulai dari depan. Hasilnya adalah (2 4 | 3 5 6 | 1 7 8) dan (5 6 | 2 3 4 |8 9 1).

2. Variasi kedua *Order Crossover*

Variasi ini merupakan modifikasi dari *Order Crossover* variasi pertama. Proses awalnya sama, yaitu menentukan 2 induk misal (2 4 3 5 6 8 9 1) dan (7 8 2 3 4 5 6 1). Kemudian menentukan titik potong, misal 2 dan 5, sehingga akan terbentuk (2 4 | 3 5 6 | 8 9 1) dan (7 8 | 2 3 4 | 5 6 1). Selanjutnya salin elemen

antara dua titik potong menjadi *offspring,* sehingga akan terbentuk (* * | 3 5 6 | * * *) dan (* * | 2 3 4 | * * *). Lalu seluruh elemen dari masing-masing induk disalin dan dihilangkan elemen yang terdapat pada masing-masing *offspring* secara bersilangan. Dilanjutkan dengan meletakkan elemen sisa ke dalam *offspring* dimulai dari paling depan. Hasilnya akan diperoleh (7 8 | 3 5 6 | 2 4 1) dan (5 6 | 2 3 4 | 8 9 1).

2.6.4 Simpel Inversion Mutation

Simpel Inversion Mutation merupakan salah satu operator yang dapat digunakan untuk menyelesaikan permasalahan TSP. Metode ini pertama kali diperkenalkan oleh Holland pada tahun 1975. Urutan langkah metode usulan Holland menurut Larrañaga et al. (1999) adalah Pertama, Memilih titik potong secara acak untuk batas atas dan batas bawah. Misal pada kromosom (1 2 3 4 5 6 7 8 9). Titik potong terpilih adalah antara elemen 2 dan 3 untuk batas atas, dan antara elemen 6 dan 7, batas atas dan bawah disimbolkan "|" (1 2 | 3 4 5 6 | 7 8 9). Kemudian elemen di antara batas atas dan bawah dibalik urutannya, sehingga hasilnya (1 2 6 5 4 3 7 8 9).

2.7 Metode K-Means dan Algoritma Genetik (GKA)

Genetic K-Means Algorithm merupakan metode gabungan dari K-Means dan Algoritma Genetik. Menurut Amirah Rahman (2014), metode K-Means dan Algoritme Genetika dapat menguraikan permasalahan MTSP menjadi TSP terpisah dengan cara memisahkan node tujuan menggunakan metode K-Means sesuai dengan jumlah agen yang tersedia, selanjutnya menyelesaikan setiap klaster TSP menggunakan algoritma genetika. Hasilnya menunjukkan bahwa metode tersebut dapat memperoleh rute yang lebih baik dibandingkan rute asli. Algoritma gabungan antara K-Means dan Algoritma Genetika ini terbukti lebih efektif dari pada hanya menggunakan Algoritma Genetika saja, karena proses *clustering* dapat membantu memecah masalah menjadi sub masalah yang pada kasus MTSP ini adalah titik kunjungan, sehingga Algoritma Genetika akan bisa lebih maksimal dalam pencarian rute pada setiap klaster (Zhanqing Lu, Kai Zhang, Juanjuan He, 2016).

2.8 Evaluasi Sistem

Evaluasi merupakan salah satu hal yang penting yang harus dilakukan dalam penelitian, karena dengan melakukan evaluasi, peneliti dapat mengetahui apakah penelitian tersebut sudah mencapai tujuan dan sesuai harapan atau belum. Menurut Agustanico Dwi Muryadi (2017), Evaluasi merupakan suatu prosedur yang dipakai untuk mengetahui serta mengukur sesuatu dengan menggunakan aturan-aturan yang sudah ditentukan sebelumnya. Evaluasi merupakan sebuah kegiatan yang bertujuan untuk mengumpulkan infomasi tentang bekerjanya sesuatu. Informasi tersebut digunakan untuk menentukan alternatif yang tepat untuk menentukan sebuah keputusan (Arikunto and Jabar, 2018).

Pada penelitian kali ini evaluasi akan digunakan untuk mengukur bahwa hasil yang diperoleh dari sistem sudah optimal atau belum. Kriteria agar dapat dikatakan optimal dari penggunaan K-Means dan Algoritma Genetika adalah ketika hasil pembagian titik kunjungan sudah hampir merata pada setiap klasternya, Jarak tempuh terpendek, dan menghindari bertabrakan jalur antar klaster yang artinya setiap klaster tidak akan mengunjungi titik yang sama (Zhanqing Lu, Kai Zhang, Juanjuan He, 2016).

2.9 Silhouette Coefficient

Silhouette Coefficient merupakan sebuah metode yang cukup popular yang digunakan untuk melihat kualitas seberapa baik suatu objek dalam klasternya. Metode ini menggabungkan nilai *cohesion* dan *separation,* nilai *cohesion* untuk melihat seberapa mirip sebuah objek dengan objek lain pada klasternya sendiri dan nilai *separation* untuk membandingkan seberapa jauh jarak dengan klaster lain (2018). Nilai Silhouette coefficient berada pada rentang -1 hingga 1. Nilai Silhouette coefficient yang mendekati 1 menunjukkan bahwa objek tersebut cocok dengan klasternya sendiri dan tidak cocok dengan klaster tetangga.

BAB 3 METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan tentang metodologi yang dilakukan dalam pengerjaan sistem yang akan diimplementasikan untuk menyelesaikan permasalahan Optimasi *Multiple Travelling Salesmen Problem* Distribusi Produk PT Indomarco Adi Prima (Stock Point Nganjuk) dengan menggunakan metode K-Means dan Algoritma Genetika (GKA) dengan data berupa jarak distribusi produk dari Gudang (Stock Point) ke toko pelanggan yang ada di wilayah Nganjuk.

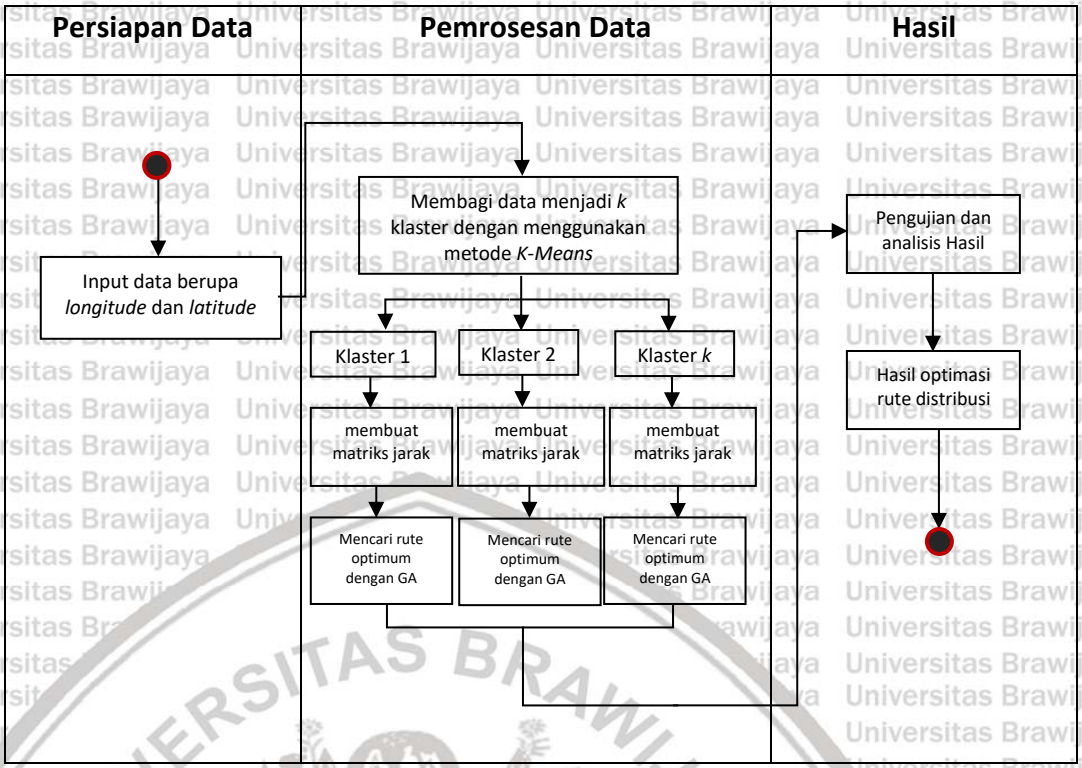
3.1 Tipe penelitian

Tipe penelitian yang digunakan pada penelitian ini adalah non-implementatif analitik yang menitikberatkan pada investigasi terhadap permasalahan tertentu yang nantinya akan menghasilkan tinjauan ilmiah. Pada umumnya, kegiatan penelitian tipe ini mengutamakan penggalian informasi dari permasalahan yang sedang diteliti dan bertujuan untuk melakukan identifikasi terhadap setiap elemen penting pada sebuah objek penelitian yang akan dijadikan sebagai dasar dalam pengambilan keputusan atau penelitian lebih lanjut. Nantinya, hasil dari kegiatan identifikasi tersebut akan mampu menjawab pertanyaan-pertanyaan penelitian yang sebelumnya sudah dirumuskan di awal.

3.2 Strategi Penelitian

Domain permasalahan pada penelitian ini termasuk ke dalam kategori *Multiple Travelling Salesman Problem*, untuk menyelesaikan permasalahan tersebut, penelitian kali ini akan menggunakan suatu sistem yang menggabungkan dua metode yaitu metode klasterisasi dan optimasi rute terpendek. Pertama yang akan dilakukan adalah melakukan klasterisasi dengan menggunakan metode K-Means. Proses yang dilakukan adalah dengan membagi wilayah kunjungan *salesman* PT Indomarco Adi Prima (Stock Point Nganjuk) sejumlah *k* klaster (sesuai dengan jumlah *salesman* harian) berdasarkan pada jarak dan letak geografis yang mencakup wilayah distribusi. Kemudian dari klaster yang sudah terbentuk akan di cari rute terpendek atau paling optimal pada setiap klaster dengan menggunakan Algoritma Genetika untuk 5 hari kerja. Selanjutnya hasil dari pencarian rute terpendek setiap *cluster* akan hitung total jarak tempuhnya, sehingga permasalahan MTSP dapat terselesaikan. Alasan diterapkannya kombinasi metode K-Means dan Algoritma Genetika pada penelitian ini adalah berdasarkan beberapa penelitian yang telah disebutkan dalam bab 2 bahwa algoritma gabungan ini sangat efektif dan akan memberikan hasil yang lebih baik dibandingkan Ketika menggunakan Algoritma Genetika saja.

Berikut alur perancangan yang digunakan untuk menyelesaikan permasalahan pada penelitian kali ini dapat dilihat pada Gambar 3.1 Diagram Blok Sistem yang terdapat 3 bagian yaitu persiapan data, pemrosesan data dan hasil.



Gambar 3.1 Diagram Blok Sistem

3.3 Subjek Penelitian

Dalam penelitian ini, yang menjadi subjek penelitian adalah PT Indomarco Adi Prima (Stock Point Nganjuk), sedangkan objek berupa data yang digunakan adalah berupa *latitude* dan *longitude*. Dalam proses pengerjaan, dari data *latitude* dan *longitude* tersebut akan diperoleh lokasi dari setiap toko. Adapun karakteristik data adalah berupa jarak distribusi produk PT Indomarco Adi Prima (Stock Point Ngajuk) yang diperoleh dari hasil perhitungan menggunakan formula haversine Persamaan 2.5. Kemudian atribut yang dimiliki oleh data adalah berupa jarak dan titik-titik dari lokasi tujuan pada setiap klaster.

3.4 Metode Pengumpulan Data

Pada tahap ini akan dilakukan proses pengumpulan data-data yang berkaitan dengan parameter rute perjalanan *salesman* dan jarak. Data diperoleh dari hasil wawancara dengan perwakilan pihak PT Indomarco Adi Prima (Stock Point Nganjuk). Data yang diperoleh antara lain adalah tentang proses kunjungan oleh *salesman* yang biasa dilakukan, serta data toko langganan yang disertai dengan titik koordinat *latitude* dan *longitude* yang nantinya titik tersebut akan digunakan dalam proses pencarian rute terpendek dengan menggunakan metode K-Means dan Algoritma Genetika. Dalam penelitian kali ini, data dipilih sesuai dengan kebutuhan yang akan digunakan pada proses perhitungan, yaitu titik *latitude* dan *longitude*.

3.5 Peralatan Pendukung

Dalam proses pengerjaan penelitian ini, kebutuhan yang diperlukan dari tahap perancangan sampai pengujian sistem terdiri dari kebutuhan perangkat keras (*hardware)* dan perangkat lunak (*software).*

3.5.1 Perangkat Keras

Kebutuhan perangkat keras yang digunakan untuk membangun sistem disesuaikan dengan spesifikasi perangkat keras yang dimiliki oleh peneliti, yaitu:

1. Laptop dengan processor Intel® Core™ i5-7200U Processor (3M Cache, up to 3.10 GHz),
2. RAM 8 GB
3. SSD 240 GB dan HDD 1 TB

3.5.2 Perangkat Lunak

Perangkat lunak yang digunakan dalam membangun sistem ini adalah sebagai berikut:

1. Sistem Operasi Windows 10 Pro 64-bit
2. Microsoft Office 2016
3. Visual Studio Code
4. Anaconda 3

BAB 4 PERANCANGAN

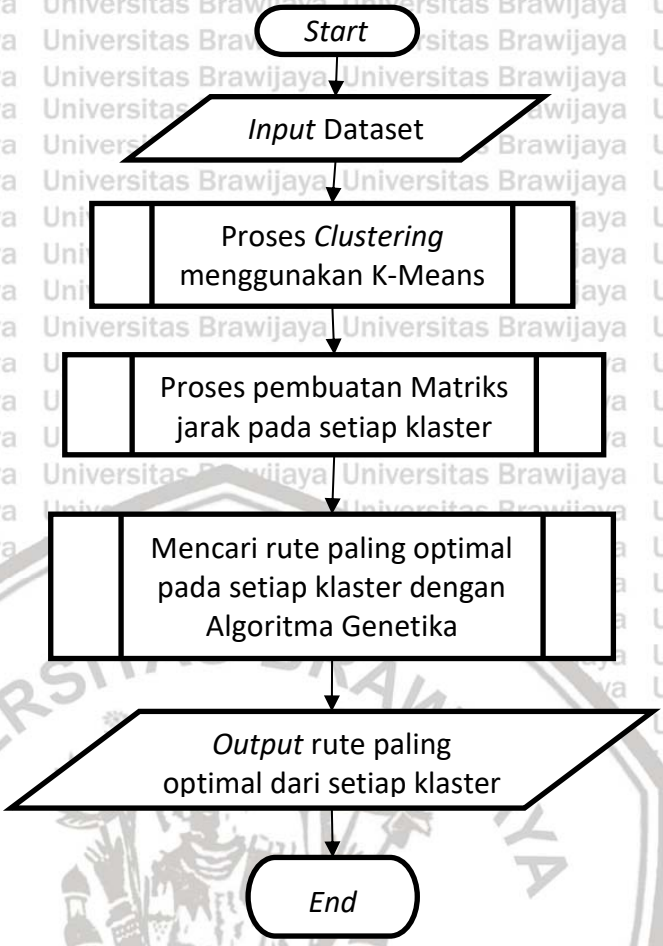
4.1 Formulasi Permasalahan

Salah satu permasalahan yang sering terjadi pada perusahaan distribusi adalah pemilihan rute yang akan dilalui, baik rute yang akan dilalui *salesman* maupun untuk proses pendistribusian barang agar dapat dilakukan dengan efektif dan efisien. Permasalahan yang akan diselesaikan pada penelitian ini terjadi pada PT Indomarco Adi Prima (*stock point* Nganjuk), yang pada perusahaan tersebut terdapat tiga *salesman* yang bertugas mengunjungi toko atau pelanggan untuk menawarkan produk dan mencatat barang pesanan pada satu hari sebelum barang didistribusikan. Pemasalahan tersebut termasuk ke dalam *Multiple Travelling Salesman Problem* (MTSP). Salah satu admin gudang PT Indomarco Adi Prima (*Stock Point* Nganjuk) mengatakan bahwa, setiap *salesman* yang bertugas hanya mengandalkan daftar toko pelanggan yang akan dikunjungi tanpa memperhatikan efektivitas jarak yang akan ditempuh, sehingga dari daftar toko yang ada masih bisa diminimumkan jarak tempuhnya.

Dari permasalahan tersebut, untuk meminimumkan jarak tempuh perlu dilakukan optimasi terhadap rute perjalanan *salesman*. Pada penelitian ini metode yang akan digunakan untuk menyelesaikan permasalahan MTSP adalah gabungan dari metode K-Means dan Algoritma Genetika. K-Means digunakan untuk melakukan pengelompokan data titik kunjungan. Data akan di pecah menjadi beberapa klaster sesuai dengan jumlah *salesman* pada perusahaan PT Indomarco Adi Prima (stock point Nganjuk), sehingga klaster tersebut akan membentuk permasalahan yang lebih kecil atau bisa disebut juga sebagai TSP. Dari hasil perolehan TSP kecil, untuk setiap klasternya akan di cari rute paling optimalnya dengan menggunakan Algoritma Genetika.

4.2 Alir Perancangan Algoritma

Pada bagian ini akan dijelaskan tentang algoritma yang akan digunakan untuk menyelesaikan permasalahan pada penilitian kali ini. Metode yang digunakan adalah gabungan dari K-Means dan Algoritma Genetika, K-Means digunakan untuk mengelompokkan titik-titik lokasi tujuan berdasarkan jarak terdekat dengan jumlah kelompok susuai dengan jumlah *salesman* yang ada. Kemudian hasil dari pengelompokan akan dicari rute paling optimal setiap kelompok untuk 5 hari kerja dengan menggunakan Algoritma Genetika. Penjelasan lebih lanjut dapat dilihat pada Gambar 4.1.



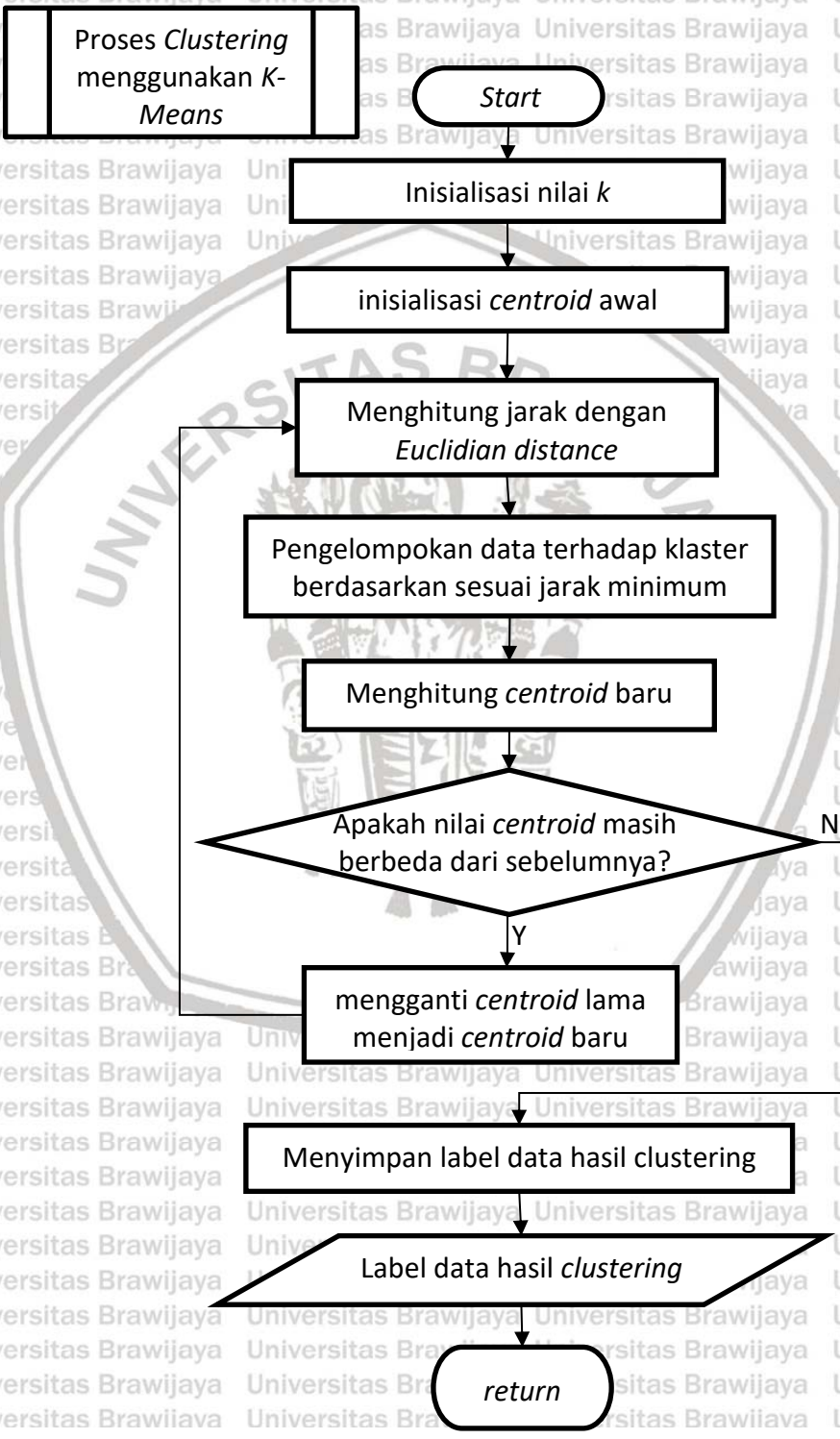
Gambar 4.1 *Flowchart* gabungan metode *K-Means* dan Algoritma Genetika

Berikut penjelasan *flowchart* pada Gambar 4.1:

1. Melakukan *input* data berupa data *latitude* dan *longitude*, data tersebut diperoleh dari data mentah dari perusahaan yang hanya diambil pada kolom *latitude* dan longitude
2. Melakukan pengelompokan data (*clustering*) dengan menggunakan K-Means. Pada proses ini, data *latitude* dan *longitude* akan dikelompokkan berdasarkan jarak terdekat antar titik sebanyak jumlah *sales* yang ada.
3. Melakukan konversi dari data *latitude* dan *longitude* menjadi matriks jarak antar titik pada setiap klaster dengan menggunakan formula haversine pada Persamaan 2.5.
4. Mencari rute paling optimal pada setiap klaster untuk 5 hari kerja dengan menggunakan Algoritma Genetika.
5. Menghasilkan *output* berupa rute paling optimal dari setiap klaster beserta total jarak tempuh.

4.2.1 Alir Proses *Clustering*

Pada tahap ini akan dilakukan pengelompokan data menggunakan K-Means berdasarkan jarak terdekat antar titik. Proses yang akan dilakukan pada tahap ini adalah inisialisasi *k* sesuai dengan jumlah *sales*, inisialisasi *centroid awal*, hingga proses penglompokan berdasarkan jarak terdekat. Untuk penjelasan lebih lanjut dapat dilihat pada Gambar 4.2.



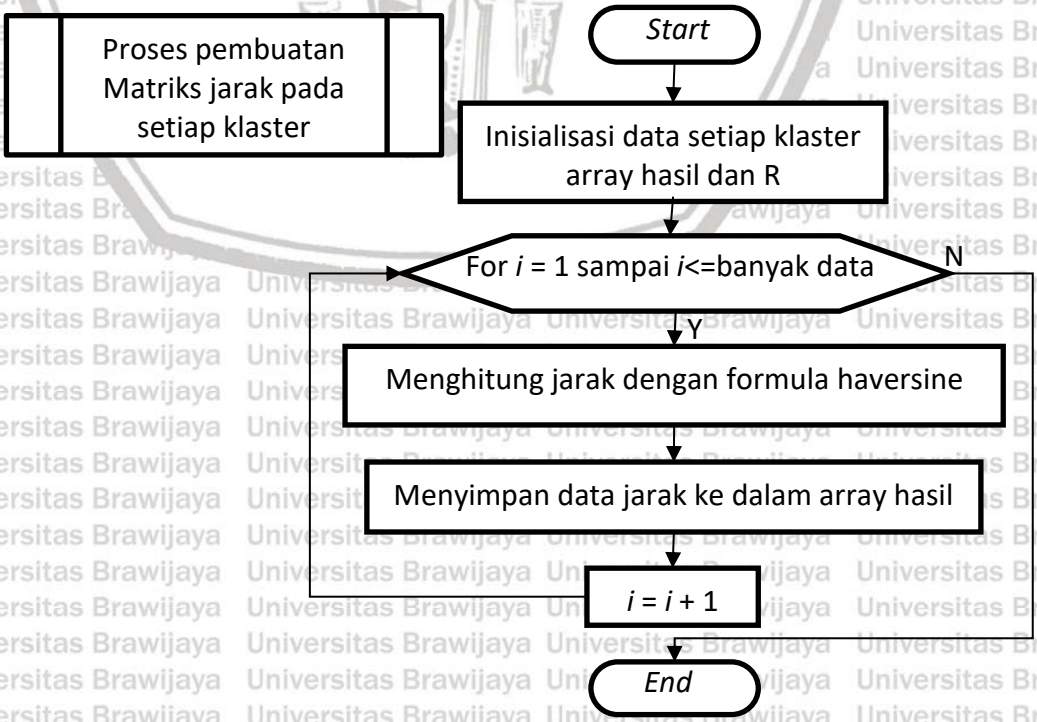
Gambar 4.2 *Flowchart* Algoritma *K-Means*

Berikut penjelasan *flowchart* pada Gambar 4.2:

1. Melakukan inisialisasi nilai *k* yang disesuaikan dengan jumlah *sales* yang ada pada PT Indomarco Adi Prima *Stock Point* Nganjuk.
2. Memilih *centroid* atau titik pusat awal dengan menggunakan pendekatan *binary search centroid*.
3. Menghitung jarak antara titik pusat dengan setiap data dengan menggunakan *Euclidian distance*.
4. Mengelompokkan data berdasarkan jarak yang terdekat dengan klaster. Data yang memiliki jarak terdekat dengan suatu klaster, maka akan dimasukkan ke dalam klaster tersebut.
5. Menghitung *centroid* baru berdasarkan data yang ada pada setiap klaster.
6. Melakukan pengecekan apakah nilai *centroid* masih berubah dan berbeda dari sebelumnya atau tidak. Jika ya maka lanjut ke langkah 6, jika nilai *centroid* tidak berubah dan sama dengan iterasi sebelumnya, maka akan langsung menuju ke langkah 8.
7. Mengganti nilai *centroid* lama dengan *centroid* baru berdasarkan hasil perhitungan sebelumnya. Dan Kembali ke langkah 3 lagi.
8. Menyimpan label data sesuai hasil klastering, yang nantinya pada setiap iterasi, data tersebut akan terus diperbaharui sesuai dengan hasil pada iterasi terbaru.
9. Hasil akhir dari *clustering,* untuk label data setiap klaster akan disimpan pada sebuah variable.

4.2.2 Alir Proses Pembuatan Matriks Jarak

Proses pembuatan matriks jarak ini akan menggunakan formula haversine. Jarak antar titik pada setiap klaster akan dihitung dengan menggunakan rumus Persamaan 2.5. Untuk penjelasan lebih lanjut dapat dilihat pada Gambar 4.3.



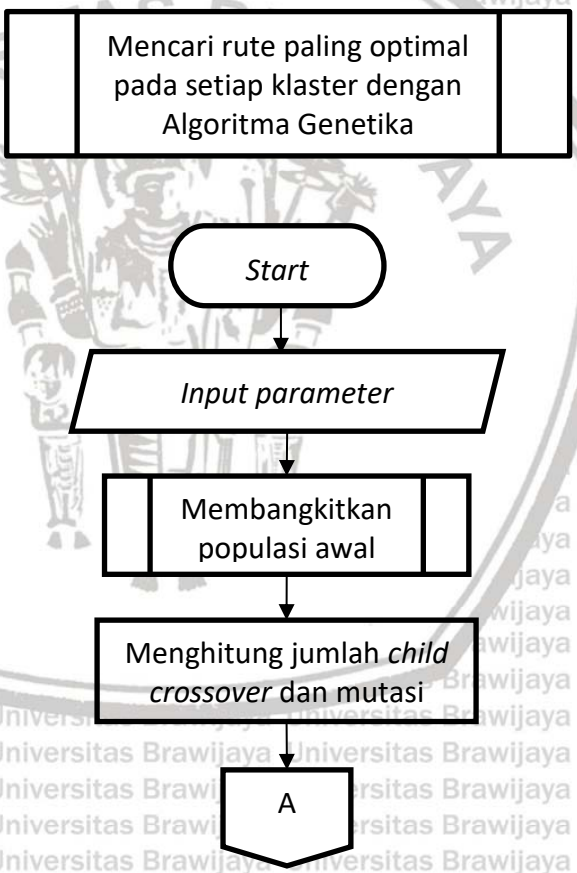
Gambar 4.3 *Flowchart* Membuat Matriks Jarak

Berikut penjelasan *flowchart* pada Gambar 4.3:

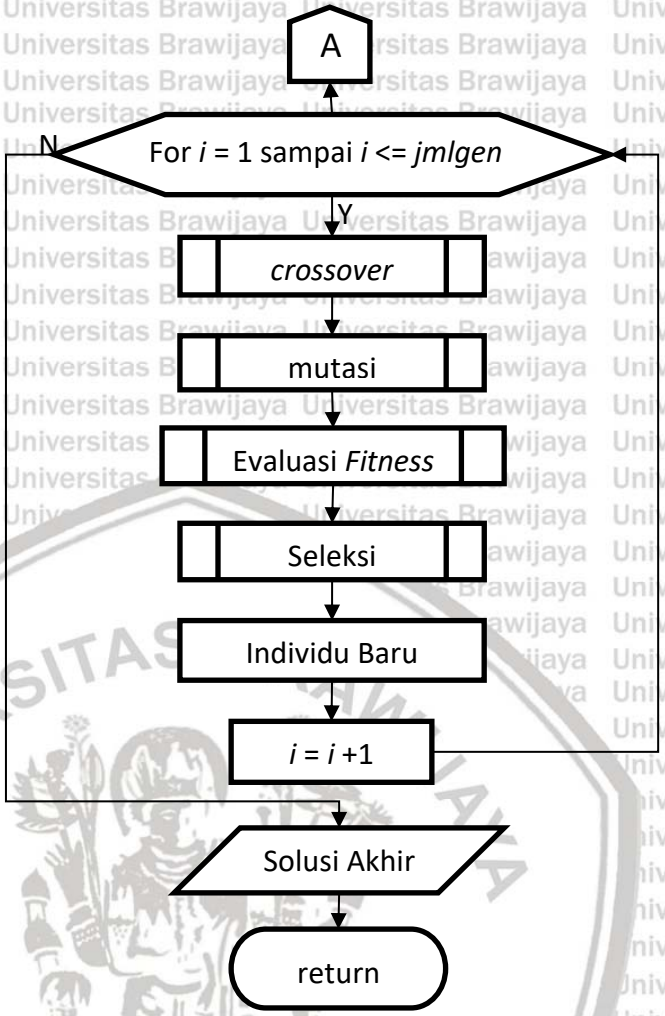
1. Melakukan Inisialisasi data setiap klaster*, array hasil* yang digunakan untuk menyimpan hasil perhitungan jarak dan inisialisasi nilai *R* sebagai radius bumi dengan nilai konstan 6371,1 untuk satuan kilometer.
2. Melakukan perulangan sejumlah banyaknya data pada tiap *cluster.*
3. Melakukan perhitungan jarak dengan menggunakan formula haversine untuk menghitung jarak setiap titik ke titik lainnya.
4. Menyimpan hasil perhitungan ke dalam *array hasil* sesuai urutan titik. jarak disimpan dalam satuan kilometer.

4.2.3 Alir Mencari rute paling optimal pada setiap klaster dengan Algoritma Genetika

Pada tahap ini akan mencari rute paling optimal pada setiap klaster dengan menggunakan Algoritma Genetika. Untuk penjelasan lebih lanjut dapat dilihat pada Gambar 4.4.



Gambar 4.4 Flowchart Algoritma Genetika



Gambar 4.4 Gambar Algoritma Genetika (Lanjutan)

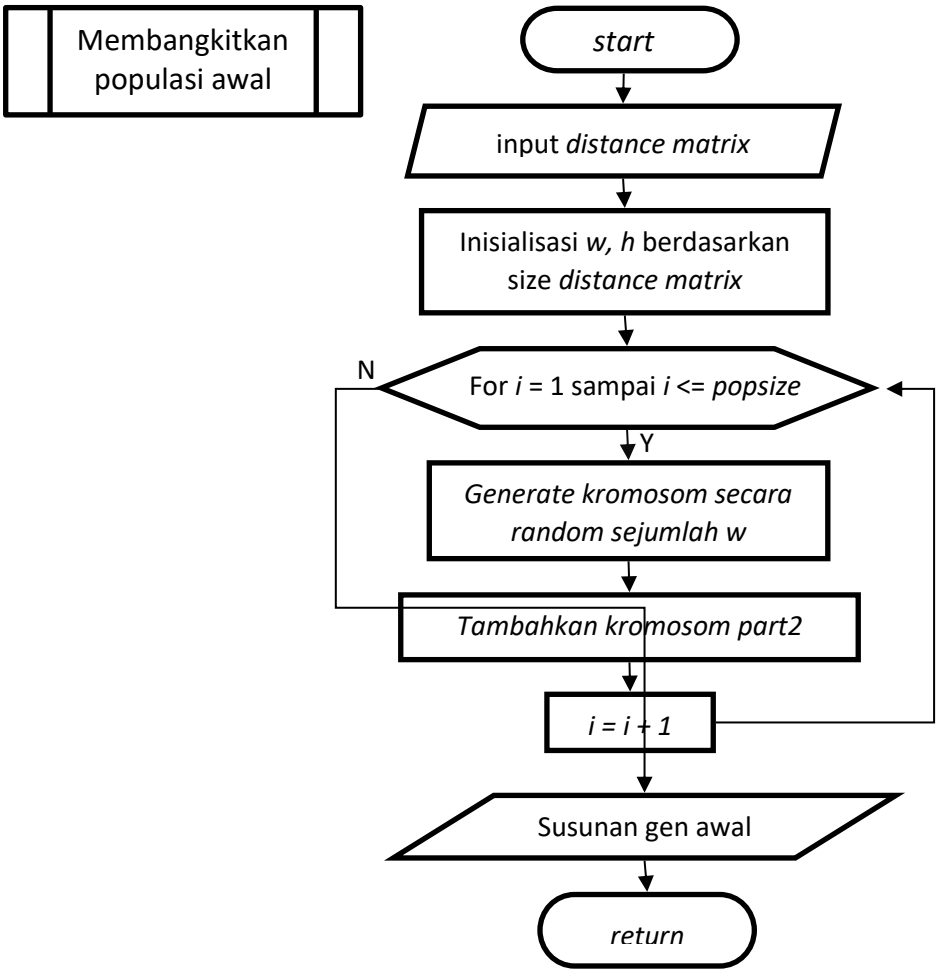
Berikut penjelasan *flowchart* pada Gambar 4.4:

1. Melakukan input beberapa parameter, yaitu data setiap klaster, *popsize* sebagai ukuran populasi setiap generasi, *jmlGen* sebagai jumlah generasi, *cr* sebagai *crossover rate* dan *mr* sebagai *mutation rate*.
2. Membangkitkan populasi awal berupa kromosom yang merepresentasikan urutan rute secara acak dari data yang ada sebanyak ukuran populasi yang telah diinputkan di awal dan setiap kromosom ditambahkan bagian 2 untuk pembagian rute untuk 5 hari kerja setiap *sales*.
3. Menghitung jumlah *child* untuk *crossover* dan mutasi yang diperoleh dari hasil perkalian antara *cr* dengan *popSize* untuk *crossover* dan *mr* dengan *popSize* untuk mutasi. Masing-masing dari keduanya akan disimpan dalam variabel *jml_childcross* dan *jml_childmut*.
4. Melakukan perulangan sebanyak jumlah gen yang telah diinputkan di awal.
5. Melakukan *crossover* dengan menggunakan metode order crossover variasi pertama. Pada proses ini akan dipilih dua individu secara acak yang akan dijadikan sebagai *parent*, kemudian menentukan operator 1 dan operator 2 untuk titik potong pertukaran kromosom antar *parent* 1 dan *parent* 2. Porses

ini nantinya akan menghasilkan individu baru atau *child,* yang merupakan hasil dari proses tukar silang antara *parent* 1 dan *parent* 2 berdasarkan operator 1 dan operator 2. Banyak *child* yang akan dihasilkan sesuai dengan *jml_childcross.*

6. Melakukan mutasi dengan menggunakan metode simple inversion mutation. Akan dipilih satu *parent* secara random, kemudian memilih dua *reverse point* pada *parent* tersebut yang nantinya akan menjadi batas atas dan batas bawah elemen yang akan dibalik urutan posisinya. Banyak *child* yang akan dihasilkan sesuai dengan *jml_childmut.*
7. Melakukan evaluasi *fitness* dengan menghitung *fitness* semua individu. Populasi awal digabungkan dengan semua *child* yang dihasilkan dari proses *crossover* dan mutasi untuk dihitung *fitness*nya.
8. Melakukan seleksi individu dengan metode *Elitism Selection*, dengan cara seluruh individu akan diurutkan secara descending berdasarkan *fitness* nya kemudian dipilih individu sejumlah ukuran populasi dari urutan atas.
9. Output berupa solusi akhir, yang merupakan solusi rute paling optimum dengan nilai *fitness* terbesar. Rute pada setiap klaster hasil optimalisasi akan disimpan dalam file berformat excel

4.2.3.1 Alir Membangkitkan Populasi Awal

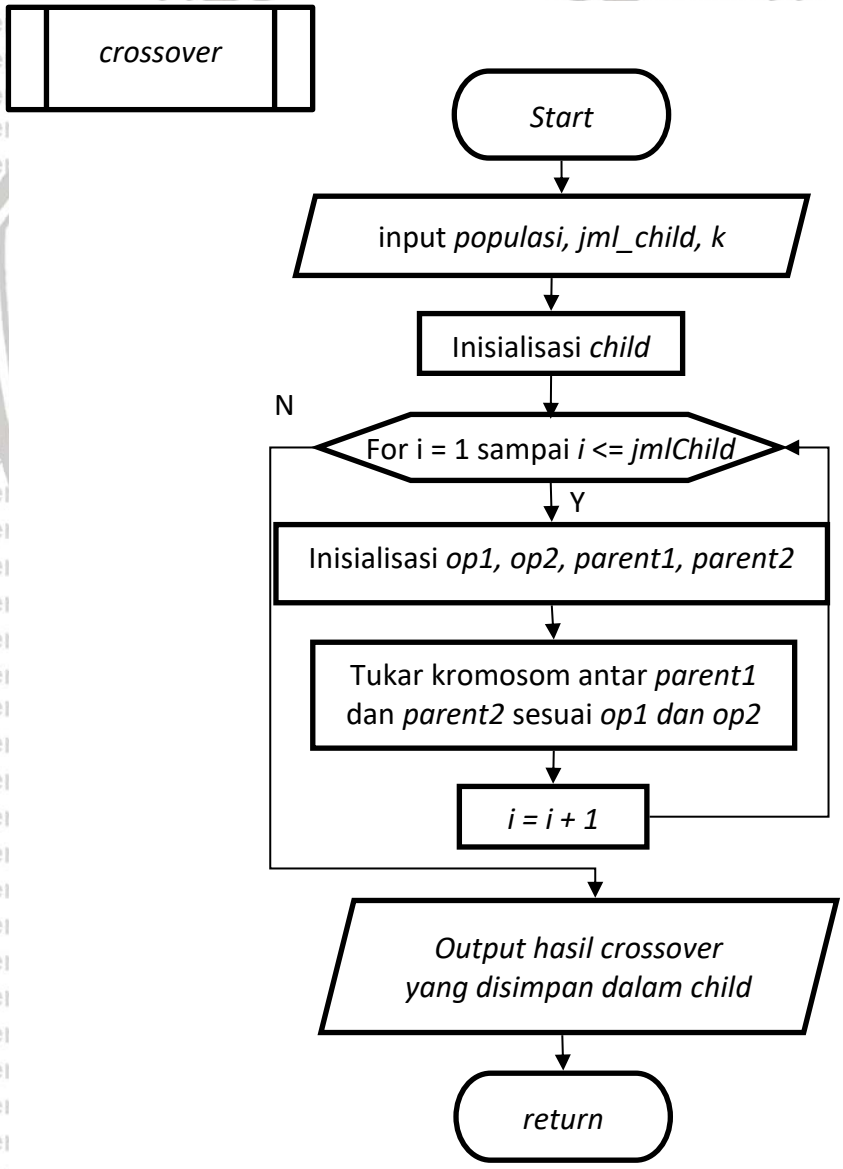


Gambar 4.5 Flowchart Membangkitkan Populasi Awal

Berikut penjelasan *flowchart* pada Gambar 4.5:

1. *input* data berupa *distance matrix* dari setiap klaster.
2. Melakukan inisialisasi terhadap variabel *w* dan *h* yang masing-masing untuk menyimpan ukuran baris dan kolom *distance matrix* yang merepresentasikan banyak anggota klaster.
3. Melakukan perulangan sebanyak *popSize.*
4. Me-*generate* kromosom secara acak yang direpresentasikan sebagai angka 1 sampai *w-1*.
5. Menambahkan kromosom *part2* yang merepresentasikan pembagian rute selama 5 hari kerja.
6. *Output* adalah berupa himpunan populasi awal yang sudah terbentuk sebanyak *popSize* dalam satu generasi.

4.2.3.2 Alir Proses *Crossover*

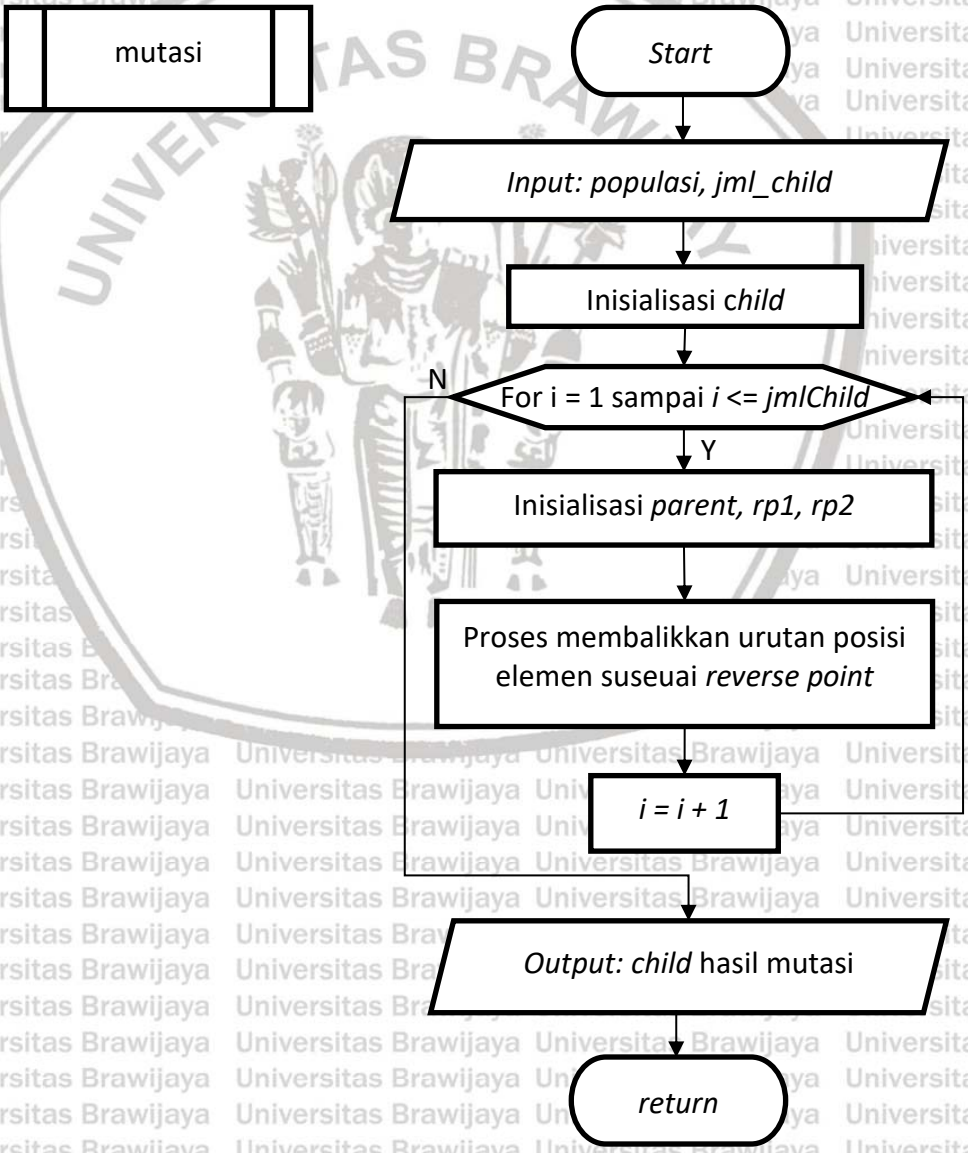


Gambar 4.6 Flowchart *Crossover*

Berikut penjelasan *flowchart* pada Gambar 4.6:

1. Input parameter: *populasi, jml_child, k.*
2. Melakukan inisialisasi *child* untuk menyimpan hasil proses *crossover*.
3. Melakukan perulangan sebanyak *jml_child.*
4. Melakukan inisialisasi *op1* sebagai operator 1, *op2* sebagai operator 2, *parent1* sebagai *parent* 1 dan *parent2* sebagai *parent* 2 yang diperoleh secara acak antara 0 sampai panjang kromosom dikurangi *k*. Hal tersebut dilakukan karena kromosom bagian 2 yang merupakan representasi pembagian rute harian tidak ikut dalam proses *crossover.*
5. Melakukan tukar silang kromosom antara *parent1* dan *parent2* yang terpilih dengan menggunakan metode order crossover berdasarkan *op1* dan *op2*. Kemudian hasil *crossover* akan ditambahkan lagi dengan kromosom bagian 2.
6. Menghasilkan susunan kromosom baru yang disimpan dalam variabel *child.*

4.2.3.3 Alir Proses Mutasi

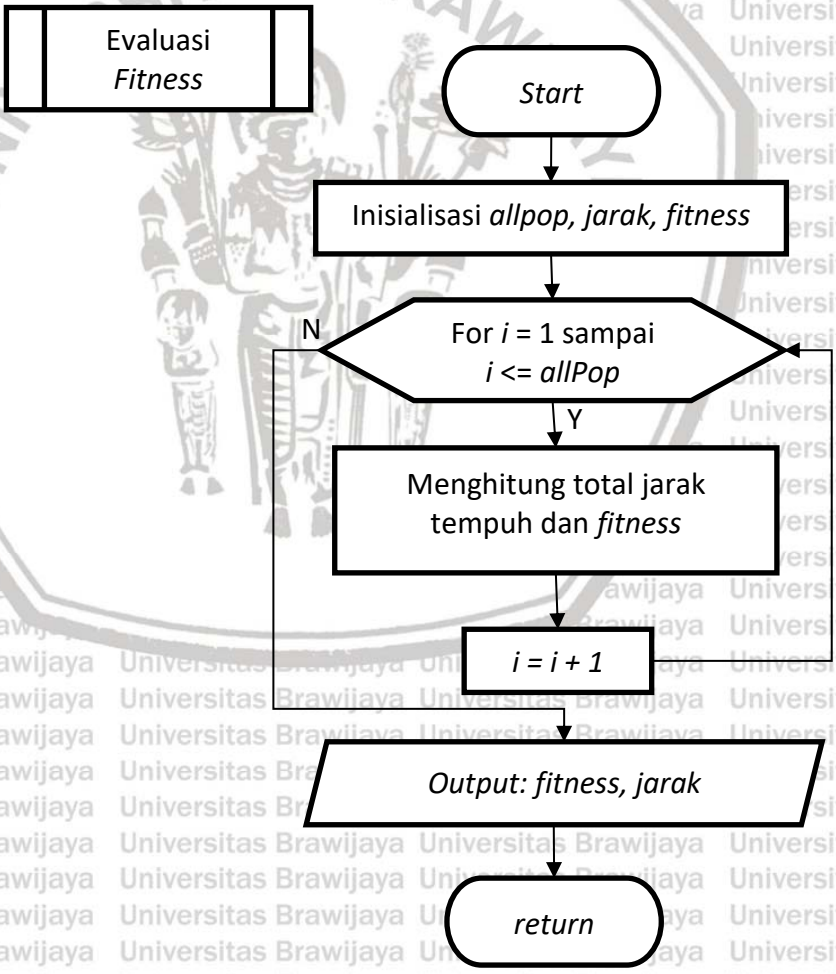


Gambar 4.7 Flowchart mutasi

Berikut penjelasan *flowchart* pada Gambar 4.7:

1. Melakukan input parameter *populasi, jml_child*
2. Melakukan inisialisasi *child*
3. Melakukan perulangan sebanyak *jml_child.*
4. *parent* merupakan *parent* teprilih yang diperoleh secara acak*, rp1* dan *rp1* merupakan indeks batas atas dan batas bawah elemen yang akan dibalik urutan posisinya yang dipilih secara acak antara 0 sampai panjang kromosom dikurangi *k*. Hal tersebut dilakukan karena kromosom bagian 2 yang merupakan representasi pembagian rute harian tidak ikut dalam proses *crossover.*
5. Melakukan mutasi kromosom menggunakan metode simple inverse mutation pada *parent* terpilih berdasarkan *rp1* dan *rp2*. Kemudian hasil mutasi digabungkan dengan kromosom bagian 2 lagi.
6. Menghasilkan susunan kromosom baru yang disimpan dalam variabel *child*.

4.2.3.4 Alir Proses Evaluasi *Fitness*



Gambar 4.8 *Flowchart* Evaluasi *fitness*

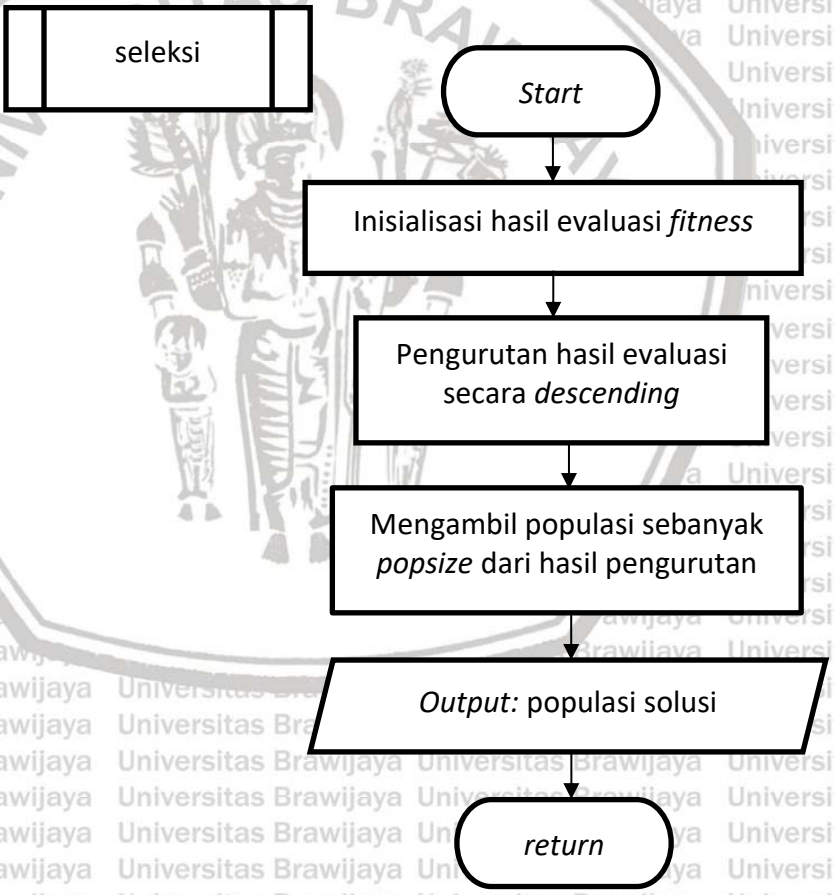
Berikut penjelasan *flowchart* pada Gambar 4.8:

1. Melakukan inisialisasi *allpop* yang merupakan gabungan *parent* dan seluruh *child* hasil dari proses *crossover* dan mutasi, *jarak* yang menyimpan hasil perhitungan jarak dari seluruh populasi, *fitness* yang menyimpan hasil perhitungan *fitness* dari seluruh populasi.
2. Melakukan perulangan sejumlah banyaknya *allpop* dikurangi *k* untuk mendapat kromosom bagian pertama saja, karena bagian 2 tidak ikut dalam proses hitung *fitness*.
3. Menghitung jarak dari satu titik ke titik lainnya sesuai pembagian untuk 5 hari kerja pada setiap populasi berdasarkan *distance matriks*. Sedangakan rumus untuk menghitung *fitness* adalah dengan menggunakan Persamaan 4.1.

$$fitness = \frac{\mathbf{100}}{\boldsymbol{total\ jarak}} \quad (4.1)$$

4. *Output* berupa hasil perhitungan jarak dan *fitness* seluruh populasi.

4.2.3.5 Alir Proses Seleksi



Gambar 4.9 Flowchart proses seleksi

Berikut penjelasan *flowchart* pada Gambar 4.9:

1. Melakukan inisialisasi hasil evaluasi *fitness.*

2. Melakukan pengurutan hasil evaluasi *fitness* secara *descending*, yaitu mulai dari yang terbesar ke yang terkecil.
3. Mengambil populasi sejumlah popsize dari hasil pengurutan dari atas.
4. *Output* berupa populasi solusi yang bisa dikatakan sebagai solusi paling baik dari setiap generasi.

4.3 Perhitungan Manualisasi

Pada tahap ini akan dijabarkan mengenai proses perhitungan manual sesuai dengan algoritma yang digunakan. Perhitungan manual yang dimaksud adalah menjelaskan langkah-langkah manual pengerjaan perhitungan dengan menggunakan metode *K-Means* dan Algoritma Genetika mulai dari proses klastering, pembuatan *distance matrix* setiap klaster, hinggan proses mencari rute paling optimal dengan menggunakan Algoritma Genetika. Data yang akan digunakan dalam proses perhitungan berjumlah 15 data dari jumlah data keseluruhan yang berisi nama toko beserta data *latitude dan longitude* lokasi toko tujuan distribusi. Namun data yang digunakan pada proses ini hanya data *latitude* dan *longitude* setiap toko saja. Data tersebut dapat dilihat pada Tabel 4.1 dan untuk data keseluruhan dapat dilihat pada Lampiran A.

Tabel 4.1 Tabel Data Sampel

No	Nama Pemilik Toko	*Latitude*	*Longitude*
1	HARSONO	-7,60100575	111,8587214
2	TOKO ABEL	-7,601423	111,903649
3	BUDAYA	-7,601373	111,904563
4	SUTINI	-7,601925933	111,8522147
5	BU JUMIATIN	-7,585797	111,9613351
6	CV. BOROBUDUR PRIMA SEJAHTERA I	-7,604983	111,900765
7	HERMIN PURWANINGTYAS	-7,569022	111,84896
8	MBAK ANIK	-7,539328	111,94946
9	TK RIZKY JAYA	-7,590789997	111,9769328
10	SAMINI	-7,527354304	111,951448
11	TK SUROSO	-7,575124598	111,8761499
12	A.YANI	-7,607466	111,899897
13	SRIATUN	-7,536208	111,94985
14	ANEKA RASA	-7,608075	111,899717
15	TOKO SLAMET	-7,58642	111,878548

4.3.1 Perhitungan K-Means

Metode K-Means digunakan untuk melakukan klastering terhadap data dengan jumlah klaster disesuaikan dengan jumlah kendaraan atau *sales* yang ada di PT indomarco Adi Prima *stock point* Nganjuk yaitu berjumlah 3. Setelah

penentuan jumlah klaster, langkah selanjutnya adalah menghitung nilai *centroid* awal menggunakan pendekatan *Binary Search* berdasarkan data pada Tabel 4.1.

Langkah pertama adalah mencari nilai minimal dan maksimal dari setiap kolom *latitude* dan *longitude*. Hasilnya adalah sebagai berikut:

max_lat = -7,527354304
min_lat = -7,608075
max_long = 111,9769328
min_long = 111,84896

Kemudian menghitung nilai *M* yang akan menyimpan nilai jarak antar *centroid* awal yang akan dibentuk dengan menggunakan Persamaan 2.3. Proses perhitungannya adalah sebagai berikut:

$$M = \frac{max\_lat - min\_lat}{k} ; \frac{max\_long - min\_long}{k}$$

$$= \frac{((-7{,}527354304) - (-7{,}608075))}{3} ; \frac{(111{,}9769328 - 111{,}84896)}{3}$$

$$= [0{,}0269069 ; 0{,}426576]$$

Selanjutnya adalah menghitung *centroid* awal dengan menggunakan Persamaan 2.4. Hasil peerhitungan *centroid* atau titik awal bisa dilihat pada Tabel 4.2. Untuk proses perhitungannya adalah sebagai berikut:

$$C_1 = \min(A_i) + (k-1)M$$

$$C_1 = (-7{,}608075) + (1-1)0{,}026906 ; (-111{,}84896) + (1-1)0{,}426576$$
$$= [-7{,}608075 ; 111{,}84896]$$

$$C_2 = (-7{,}608075) + (2-1)0{,}026906 ; (-111{,}84896) + (2-1)0{,}426576$$
$$= [-7{,}581168101; 111{,}8916176]$$

$$C_3 = (-7{,}608075) + (3-1)0{,}026906 ; (-111{,}84896) + (3-1)0{,}426576$$
$$= [-7{,}554261203 ; 111{,}9342752]$$

Tabel 4.2 *Centroid* Awal

	Latitude	***Longitude***
C1	-7,608075	111,84896
C2	-7,581168101	111,8916176
C3	-7,554261203	111,9342752

Keterangan penjelas:
C1 = *centroid* klaster 1
C2 = *centroid* klaster 2
C3 = *centroid* klaster 3

4.3.1.1 *Proses Perhitungan Jarak Data Terhadap Centroid*

Pada tahap ini akan melakukan perhitungan jarak data pada terhadap *centroid* pada setiap klaster. Perhitungan jarak yang dilakukan adalah dengan menggunakan rumus *Euclidian Distance* seperti yang terdapat pada Persamaan 2.1, di mana x_i adalah titik koordinat objek ke-i dan y_i adalah titik koordinat *centroid* ke-i. Berikut adalah contoh perhitungan jarak terhadap *centroid* pada setiap *cluster*.

1. Perhitungan jarak data pada baris pertama terhadap *centroid* 1.

$$d(x = (-7{,}60100575\ ,111{,}8587214), y = (-7{,}608075,\ \ 111{,}84896)) = \sqrt{(-7{,}60100575 - (-7{,}608075))^2 + (111{,}8587214 - 111{,}84896)^2} = 0{,}012052354$$

2. Perhitungan jarak data pada baris pertama terhadap *centroid* 2.

$$d(x = (-7{,}60100575\ ,111{,}8587214), y = (-7{,}581168101, 111{,}8916176)) = \sqrt{(-7{,}60100575 - (-7{,}581168101))^2 + (111{,}8587214 - 111{,}8916176)^2} = 0{,}03841474$$

3. Perhitungan jarak data pada baris pertama terhadap *centroid* 3.

$$d(x = (-7{,}60100575\ ,111{,}8587214), y = (-7{,}554261203,\ \ 111{,}9342752)) = \sqrt{(-7{,}60100575 - (-7{,}554261203))^2 + (111{,}8587214 - 111{,}9342752)^2} = 0{,}088844974$$

Langkah 1 sampai 3 berlaku untuk baris ke dua dan seterusnya, yaitu untuk menghitung jarak data setiap baris terhadap masing-masing *centroid*. Jika semua data sudah mendapatkan hasil perhitungan jarak, maka langkah selanjutnya adalah mencari nilai paling minimum jarak data setiap baris terhadap ke tiga *centroid* kemudian memasukkan data pada baris tersebut ke dalam klaster dengan jarak terhadap *centroid* yang paling minimal. Hasil perhitungan jarak dan pengelompokan dapat dilihat pada Tabel 4.3.

Tabel 4.3 Perhitungan Iterasi ke-0

No	*Latitude*	*Longitude*	c1	c2	c3	jara terpendek	Kelas
1	-7,60100575	111,8587214	0,012052	0,038414	0,088845	0,012052	1
2	-7,601423	111,903649	0,055092	0,023558	0,056233	0,023558	2
3	-7,601373	111,904563	0,056005	0,023996	0,055698	0,023996	2
4	-7,601925933	111,8522147	0,006957	0,044536	0	0	1
5	-7,585797	111,9613351	0,114562	0,069871	0,041554	0,041554	3

6	-7,604983	111,900765	0,051897	0,025511	0,060791	0,025511	2
7	-7,569022	111,84896	0,039053	0,044353	0,086582	0,039053	1
8	-7,539328	111,94946	0,121763	0,071388	0,021297	0,021297	3
9	-7,590789997	111,9769328	0,129134	0,085856	0,056160	0,056160	3
10	-7,527354304	111,951448	0,130459	0,080471	0,031920	0,031919	3
11	-7,575124598	111,8761499	0,042720	0,016606	0,061756	0,016606	2
12	-7,607466	111,899897	0,050940	0,027570	0,063345	0,027570	2
13	-7,536208	111,94985	0,123869	0,073569	0,023843	0,023843	3
14	-7,608075	111,899717	0,050757	0,028099	0,063954	0,028099	2
15	-7,58642	111,878548	0,036665	0,014085	0,064340	0,014085	2

4.3.1.2 Proses Menghitung *Centroid* Baru

Langkah selanjutnya adalah menghitung *centroid* baru dengan cara menghitung rata-rata data yang terdapat setiap klaster dengan menggunakan rumus pada Persamaan 2.2. Berikut adalah contoh perhitungan untuk mencari *centroid* baru setiap *cluster*.

$$\mu_j = \frac{1}{N_j} \sum_{j \in S_j} x_i$$

1. Menghitung *centroid* baru kolom *latitude* pada C1

$$\mu_j = \frac{1}{3} \times \left(-7{,}60100575 + (-7{,}601925933) + (-7{,}569022)\right)$$
$$= -7{,}590651228$$

2. Menghitung *centroid* baru kolom *longitude* pada C1

$$\mu_j = \frac{1}{3} \times (111{,}8587214 + 111{,}8522147 + 111{,}84896)$$
$$= 111{,}8532987$$

Langkah 1 dan 2 juga berlaku untuk perhitungan mencari *centroid* baru pada klaster 1 (C1) dan klaster 2 (C2). Jika semua proses mencari *centroid* baru pada setiap klaster sudah selesai, maka hasilnya bisa dilihat pada Tabel 4.4.

Tabel 4.4 Hasil Perhitungan *Centroid* Baru Iterasi ke- 0

	Latitude	*Longitude*
C1	-7,590651228	111,8532987
C2	-7,5978378	111,8947556
C3	-7,55589546	111,9578052

4.3.1.3 Proses Menghitung Jarak Data Terhadap *Centroid* Baru

Proses menghitung jarak data terhadap *centroid* Kembali dilakukan, namun pada tahap ini *centroid* yang digunakan bukan *centroid* awal lagi, melainkan *centroid* baru pada Tabel 4.4 yang dihasilkan dari proses sebelumnya.

Untuk langkah-langkah perhitungannya sama dengan proses menghitung jarak data terhadap *centroid* di atas. Hasil perhitungan pada tahap ini dapat dilihat pada Tabel 4.5.

Tabel 4.5 Perhitungan Iterasi ke-1

Terlihat pada Tabel 4.5, masih terdapat data yang berpindah klaster, yaitu pada data ke-11 berpindah dari klaster 2 ke klaster 1 pada baris berwarna merah. Langkah selanjutnya menghitung *centroid* baru dan langkah-langkahnya masih

No	*Latitude*	*Longitude*	c1	c2	c3	jara terpendek	kelas Lama	kelas Baru
1	-7,60100575	111,8587214	0,011688	0,036173	0,108869	0,011688	1	1
2	-7,601423	111,903649	0,051489	0,009588	0,070750	0,009588	2	2
3	-7,601373	111,904563	0,052373	0,010425	0,070021	0,010425	2	2
4	-7,60192593	111,8522147	0,011326	0,042736	0,115187	0,011326	1	1
5	-7,585797	111,9613351	0,108145	0,067659	0,030109	0,030109	3	3
6	-7,604983	111,900765	0,049582	0,009336	0,075254	0,009336	2	2
7	-7,569022	111,84896	0,022060	0,054107	0,109633	0,022060	1	1
8	-7,539328	111,94946	0,109000	0,080099	0,018550	0,018550	3	3
9	-7,59079	111,9769328	0,123634	0,082478	0,039793	0,039793	3	3
10	-7,5273543	111,951448	0,116789	0,090454	0,029240	0,029240	3	3
11	-7,5751246	111,8761499	0,027627	0,029360	0,083888	0,027627	2	1
12	-7,607466	111,899897	0,049539	0,010915	0,077542	0,010914	2	2
13	-7,536208	111,94985	0,110843	0,082665	0,021234	0,021233	3	3
14	-7,608075	111,899717	0,049580	0,011376	0,078082	0,011376	2	2
15	-7,58642	111,878548	0,025601	0,019825	0,084932	0,019825	2	2

sama dengan proses menghitung *centroid* baru sebelumnya. Hasil penghitungan untuk mencari *centroid* baru dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil Perhitungan *Centroid* Baru Iterasi ke- 1

	Latitude	*Longitude*
C1	-7,58676957	111,8590115
C2	-7,601623333	111,8978565
C3	-7,55589546	111,9578052

Hasil perhitungan dari *centroid* baru di atas dicek apakah masih berbeda dengan *centroid* sebelumnya atau tidak. Karena antara *centroid* baru dan lama masih berbeda, maka akan dilanjutkan ke iterasi ke-2, dimana hal pertama yang dilakukan adalah menghitung jarak setiap data terhadap *centroid* setiap klaster. *Centroid* yang digunakan adalah *centroid* terbaru pada Tabel 4.6. Untuk langkah-langkah perhitungannya sama dengan proses menghitung jarak data terhadap *centroid* di atas. Hasil perhitungan pada tahap ini dapat dilihat pada Tabel 4.7.

Tabel 4.7 Perhitungan Iterasi ke-2

No	*Latitude*	*Longitude*	c1	c2	c3	jara terpendek	kelas Lama	kelas Baru
1	-7,60100575	111,8587214	0,014239	0,039140	0,108869	0,014239	1	1
2	-7,601423	111,903649	0,046981	0,005796	0,070750	0,005795	2	2
3	-7,601373	111,904563	0,047835	0,006711	0,070021	0,00671	2	2
4	-7,601925933	111,8522147	0,016610	0,045642	0,115187	0,016610	1	1
5	-7,585797	111,9613351	0,102328	0,065421	0,030109	0,030109	3	3
6	-7,604983	111,900765	0,045553	0,004443	0,075254	0,004443	2	2
7	-7,569022	111,84896	0,02039	0,058768	0,109633	0,020396	1	1
8	-7,539328	111,94946	0,102135	0,080892	0,018550	0,018550	3	3
9	-7,590789997	111,9769328	0,117989	0,079814	0,039793	0,039793	3	3
10	-7,527354304	111,951448	0,109884	0,091585	0,029240	0,029240	3	3
11	-7,575124598	111,8761499	0,020720	0,034254	0,083888	0,020720	1	1
12	-7,607466	111,899897	0,045825	0,006188	0,077542	0,006188	2	2
13	-7,536208	111,94985	0,103962	0,083561	0,021234	0,021233	3	3
14	-7,608075	111,899717	0,045944	0,006714	0,078082	0,006714	2	2
15	-7,58642	111,878548	0,019539	0,024575	0,084932	0,019539	2	1

Dilihat dari Tabel 4.7, masih terdapat data yang berpindah klaster, yaitu pada data ke-15 berpindah dari klaster 2 ke klaster 1 pada baris berwarna merah. Proses selanjutnya adalah menghitung *centroid* baru dan langkah-langkahnya masih sama dengan proses menghitung *centroid* baru sebelumnya. Hasil penghitungan untuk mencari *centroid* baru dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil Perhitungan *Centroid* Baru Iterasi ke- 2

	Latitude	*Longitude*
C1	-7,586699656	111,8629188
C2	-7,604664	111,9017182
C3	-7,55589546	111,9578052

Hasil perhitungan dari *centroid* baru di atas dicek apakah masih berbeda dengan *centroid* sebelumnya atau tidak. Karena antara *centroid* baru dan lama masih berbeda, makan akan masuk ke iterasi ke-3, dimana hal pertama yang dilakukan adalah menghitung jarak setiap data terhadap *centroid* setiap klaster. *Centroid* yang digunakan adalah *centroid* terbaru pada Tabel 4.8. Untuk langkah-langkah perhitungannya sama dengan proses menghitung jarak data terhadap *centroid* di atas. Hasil perhitungan pada tahap ini dapat dilihat pada Tabel 4.9.

Tabel 4.9 Perhitungan Iterasi ke-3

No	*Latitude*	*Longitude*	c1	c2	c3	jara terpendek	kelas Lama	kelas Baru
1	-7,601006	111,8587214	0,014909	0,043152	0,108869	0,014909	1	1

2	-7,601423	111,903649	0,043309	0,003772	0,070750	0,003772	2	2
3	-7,601373	111,904563	0,044153	0,004350	0,070021	0,004350	2	2
4	-7,601926	111,8522147	0,018612	0,049579	0,115187	0,018612	1	1
5	-7,585797	111,9613351	0,098420	0,062531	0,030109	0,030109	3	3
6	-7,604983	111,900765	0,042031	0,001005	0,075254	0,001005	2	2
7	-7,569022	111,84896	0,022524	0,063669	0,109638	0,022524	1	1
8	-7,539328	111,94946	0,098658	0,080920	0,018550	0,018550	3	3
9	-7,59079	111,9769328	0,114087	0,076483	0,039793	0,039793	3	3
10	-7,527354	111,951448	0,106579	0,091923	0,029240	0,029240	3	3
11	-7,575125	111,8761499	0,017579	0,039068	0,083888	0,017579	1	1
12	-7,607466	111,899897	0,042410	0,003341	0,077542	0,003341	2	2
13	-7,536208	111,94985	0,100530	0,083683	0,021234	0,021233	3	3
14	-7,608075	111,899717	0,042556	0,003954	0,078082	0,003954	2	2
15	-7,58642	111,878548	0,015631	0,029490	0,084932	0,015631	1	1

Jika dilihat dari Tabel 4.9, sudah tidak ada lagi data yang berpindah klaster. Langkah selanjutnya adalah melakukan perhitungan *centroid* baru, untuk langkah perhitungannya sama seperti sebelumnya. Hasil penghitungan untuk mencari *centroid* baru dapat dilihat pada Tabel 4.10.

Tabel 4.10 Hasil Perhitungan *Centroid* Baru Iterasi ke- 3

	Latitude	*Longitude*
c1	-7,55589546	111,9578052
c2	-7,604664	111,9017182
c3	-7,586699656	111,8629188

Hasil perhitungan dari *centroid* baru di atas dicek apakah masih berbeda dengan *centroid* sebelumnya atau tidak. Karena antara *centroid* baru dan lama sudah sama, itu artinya iterasi akan berhenti. Untuk setiap klalster akan ditambah satu titik lagi, yaitu titik *latitude* dan *longitude* lokasi PT Indomarco Adi Prima (*Stock Point* Nganjuk), karena titik setiap klaster ada di lokasi tersebut. Untuk hasil klaster dapat dilihat pada Tabel 4.11.

Tabel 4.11 Hasil Akhir Klasterisasi

class	Nama Toko	*Latitude*	*Longitude*
1	Gudang PT Indomarco	-7,616966	111,894101
	HARSONO	-7,60100575	111,8587214
	SUTINI	-7,601925933	111,8522147
	HERMIN PURWANINGTYAS	-7,569022	111,84896
	TK SUROSO	-7,575124598	111,8761499
	TOKO SLAMET	-7,58642	111,878548
2	Gudang PT Indomarco	-7,616966	111,894101
	TOKO ABEL	-7,601423	111,903649

	BUDAYA	-7,601373	111,904563
	CV, BOROBUDUR PRIMA SEJAHTERA I	-7,604983	111,900765
	A,YANI	-7,607466	111,899897
	ANEKA RASA	-7,608075	111,899717
3	Gudang PT Indomarco	-7,616966	111,894101
	BU JUMIATIN	-7,585797	111,9613351
	MBAK ANIK	-7,539328	111,94946
	TK RIZKY JAYA	-7,590789997	111,9769328
	SAMINI	-7,527354304	111,951448
	SRIATUN	-7,536208	111,94985

4.3.1.4 Perhitungan *Silhouette Coefficient*

Pada bagian ini akan dilakukan perhitungan untuk mengetahui nilai *silhouette* dari klaster yang sudah terbentuk. Perhitungan dilakukan dengan menggunakan Persamaan 4.1 berikut:

$$s_i = \frac{(a_i - b_i)}{\max(a_i, b_i)} \quad (4.1)$$

a_i = rata-rata jarak objek ke-*I* ke objek lain dalam satu klaster.

b_i = rata-rata jarak objek ke-*i* ke objek lain pada klaster lain.

Titik gudang PT Indomarco Adi Prima *(Stock Point Nganjuk)* tidak termasuk dalam perhitungan nilai *silhouette* karena merupakan titik pusat keberangkatan, jadi tidak ikut dalam proses *K-Means*. Perhitungan jarak menggunakan rumus *euclidian*. Contoh perhitungan dilakukan pada titik satu pada klaster 1 (toko Harsono).

1. Perhitungan toko Harsono ke toko lain dalam klaster 1 (a_1).

{Harsono -> Sutini} =

$$\sqrt{\left(((-7{,}601925933) - (-7{,}60100575))^2 + (111{,}8522147 - 111{,}8587214)^2\right)} = 0{,}006571$$

{Harsono -> HERMIN PURWANINGTYAS} =

$$\sqrt{(((-7{,}569022) - (-7{,}60100575))^2 + (111{,}84896 - 111{,}8587214)^2)} = 0{,}033440$$

{Harsono -> TK SUROSO} =

$$\sqrt{(((-7{,}575124598) - (-7{,}60100575))^2 + (111{,}8761499 - 111{,}8587214)^2)} = 0{,}031202$$

{Harsono -> TOKO SLAMET} =

$$\sqrt{(((-7{,}58642) - (-7{,}60100575))^2 + (111{,}878548 - 111{,}8587214)^2)} = 0{,}024613$$

Kemudian seluruh hasilnya dicari rata-ratanya.

$$\frac{(0{,}006571 + 0{,}033440 + 0{,}031202 + 0{,}024613)}{4} = 0{,}023956$$

$a_1 = 0{,}023956$

2. Perhitungan toko Harsono ke toko lain dalam klaster 2 dan 3 (b_1). Untuk proses perhitungannya sama dengan langkah 1, sehingga akan langsung diperoleh hasilnya

   Klaster 2

   {Harsono -> TOKO ABEL} = 0,044929
   {Harsono -> BUDAYA} = 0,045843
   {Harsono -> CV. BOROBUDUR PRIMA SEJAHTERA I} = 0,042231
   {Harsono -> A.YANI} = 0,041679
   {Harsono -> ANEKA RASA} = 0,041600
   Nilai rata-rata = 0,043256

   Klaster 3

   {Harsono -> BU JUMIATIN} = 0,103734
   {Harsono -> MBAK ANIK} = 0,109716
   {Harsono -> TK RIZKY JAYA} = 0,118651
   {Harsono -> SAMINI} = 0,118417
   {Harsono -> SRIATUN} = 0,111817
   Nilai rata-rata = 0,112467

   Selanjutnya nilai rata-rata antara klaster 2 dan 3 dicari yang paling minimum untuk b_1.

$$b_1 = 0{,}043256$$

3. Menghitung nilai *silhouette* pada titik Harsono menggunakan Persamaan 4.1.

$$s_i = \frac{(0{,}043256 - 0{,}023956)}{0{,}043256} = 0{,}446169$$

Langkah 1 sampai 3 juga dilakukan pada semua titik pada setiap klaster kemudian seluruh hasil s_i pada seluruh titik dihitung rata-ratanya untuk mendapatkan nilai *silhouette*. Hasil dari keseluruhan perhitungan diperoleh nilai *silhouette* dari klaster yang terbentuk adalah 0,588743.

4.3.2 Membuat *Distance Matrix*

Setelah klaster terbentuk, langkah selanjutnya adalah membuat *distance matrix* dari data *latitude* dan *longitude*. Dari hasil klastering pada Tabel 4.10 terdapat 3 klaster dengan tiap klaster memiliki 5 anggota. Pada tahap ini akan dicari jarak antar data berdasarkan fitur *latitude* dan *longitude* untuk semua data pada masing-masing klaster dengan menggunakan formula *haversine,* sehingga nanti akan terbentuk sebuah *distance matrix*. Berikut adalah contoh perhitungan menggunakan Persamaan 2.5.

- Contoh perhitungan jarak pada klaster 3 dari toko Bu Jumiatin ke toko Mbak Anik

Latitude	*Longitude*	Nama Toko
-7,585797	111,9613351	BU JUMIATIN
-7,539328	111,94946	MBAK ANIK

$$d = 2r.\arcsin\left\{\sqrt{sin^2\left(\frac{Lat_1 - Lat_2}{2}\right) + cos(Lat_1).\cos(Lat_2).sin^2\left(\frac{Long_1 - Long_2}{2}\right)}\right\}$$

1) Titik asal (Toko Bu Jumiatin)
   latitude 1 = -7,585797 * $\frac{\pi}{180}$ = -0,132397134 Radian
   longitude 1 = 111,9613351 * $\frac{\pi}{180}$ = 1,95409393 Radian
2) Titik tujuan (Toko Mbak Anik)
   latitude 1 = -7,539328 * $\frac{\pi}{180}$ = -0,131586097 Radian
   longitude 1 = 111,94946 * $\frac{\pi}{180}$ = 1,95388667 Radian
3) $Lat_1 - Lat_2 = (-0{,}132397134) - (-0{,}131586097) = -0{,}000811037$
4) $Long_1 - Long_2 = 1{,}95409393 - 1{,}95388667 = 0{,}00020726$
5) $\alpha = sin^2\left(\frac{Lat_1 - Lat_2}{2}\right) + cos(Lat_1).\cos(Lat_2).sin^2\left(\frac{Long_1 - Long_2}{2}\right)$
   $= sin^2\left(\frac{-0{,}000811037}{2}\right)$
   $+ cos(-0{,}132397134).\cos(-0{,}131586097).sin^2\left(\frac{0{,}00020726}{2}\right)$
   = 0,000000174998408
6) $c = 2 * \arcsin(\sqrt{a})$
   $= 2 * \arcsin(\sqrt{0{,}000000174998408})$
   $= 0{,}000836656$
7) $d = R * c$
   $= 5{,}33042028845869$

Hasil penyusunan *distance matrix* bisa dilihat pada Tabel 4.12 untuk klaster 1, Tabel 4.13 untuk klaster 2, dan Tabel 4.14 untuk klaster 3.

Tabel 4.12 *Distance Matrix* Klaster 1

	Gudang PT Indomarco	HARSONO	SUTINI	HERMIN P.	TK SUROSO	TOKO SLAMET
Gudang PT Indomarco	0	4,284319	4,910187	7,292296	5,055859	3,804682
HARSONO	4,284319	0	0,724427	3,715674	3,460146	2,721420
SUTINI	4,910187	0,724427	0	3,676353	3,980179	3,375996
HERMIN P.	7,292296	3,715674	3,676353	0	3,072923	3,791979
TK SUROSO	5,055859	3,460146	3,980179	3,072923	0	1,283524
TOKO SLAMET	3,804682	2,721420	3,375996	3,791979	1,283524	0

Tabel 4.13 *Distance Matrix* Klaster 2

	Gudang PT Indomarco	TOKO ABEL	BUDAYA	CV. BOROBUDUR PRIMA SEJAHTERA I	A.YANI	ANEKA RASA
Gudang PT Indomarco	0	2,023506	2,082307	1,521494	1,234504	1,166430
TOKO ABEL	2,023506	0	0,100893	0,507688	0,789017	0,857289
BUDAYA	2,082307	0,100893	0	0,579977	0,850599	0,916878
CV, BOROBUDUR PRIMA SEJAHTERA I	1,521494	0,507688	0,579977	0	0,292206	0,362704
A,YANI	1,234504	0,789017	0,850599	0,292206	0	0,070565
ANEKA RASA	1,166430	0,857289	0,916878	0,362704	0,070565	0

Tabel 4.14 *Distance Matrix* Klaster 3

	Gudang PT Indomarco	BU JUMIATIN	MBAK ANIK	TK RIZKY JAYA	SAMINI	SRIATUN
Gudang PT Indomarco	0	8,180956	10,57186	9,582389	11,80038	10,881248
BU JUMIATIN	8,180956	0	5,330420	1,806649	6,589388	5,657597
MBAK ANIK	10,571861	5,330420	0	6,474292	1,349350	0,349587
TK RIZKY JAYA	9,5823892	1,806649	6,474292	0	7,592642	6,763797
SAMINI	11,800386	6,589388	1,349350	7,592642	0	1,000137
SRIATUN	10,881248	5,657597	0,349587	6,763797	1,000137	0

4.3.3 Mencari Rute Paling Optimal Setiap Klaster

Pada tahap ini metode yang digunakan dalam mencari rute paling optimal adalah dengan menggunakan Algoritma Genetika. Data *distance matrix* yang telah dibuat pada proses sebelumnya akan digunakan pada tahap ini, di mana terdapat 3 *distance matrix* yang disesuaikan dengan jumlah klaster. Dari *distance matrix* tersebut nantinya akan dicari rute dengan jarak terpendek menggunakan Algoritma Genetika. Hal pertama yang dilakukan pada tahap ini adalah melakukan inisialisasi distance matrix semua klaster seperti pada Tabel 4.12, Tabel 4.13, dan Tabel 4.14. Kemudian juga dilakukan inisialisasi parameter yang berupa *popSize, Crossover rate, mutation rate dan jmlGen*. Untuk lebih detailnya bisa dilihat pada Tabel 4.15.

Tabel 4.15 Tabel Parameter

PopSize	3

cr	0,5
mr	0,5
jmlGen	2

4.3.3.1 Inisialisasi

Pada tahap ini akan dilakukan inisialisasi kromosom. Setiap populasi memiliki satu kromosom yang tersusun dari dua bagian, bagian pertama adalah urutan rute perjalanan distribusi yang direpresentasikan sebagai angka 1 sampai 5 secara acak, sedangkan bagian dua merepresentasikan pembagian titik kunjungan harian. Jumlah anggota setiap klaster adalah 6 termasuk Gudang PT Indomarco, namun titik lokasi Gudang PT Indomarco hanya digunakan untuk titik keberangkatan, sehingga tidak perlu dimasukkan ke dalam kromosom. Kromosom diinisialisasi sebanyak jumlah *popSize,* yaitu 3 populasi. Contoh perhitungan manual yang akan dibuat adalah hanya untuk 2 hari kerja. Untuk lebih jelas mengenai hasil inisialisasi bisa dilihat pada Tabel 4.16.

Tabel 4.16 Inisialisasi *Chromosome*

No	Chromosome
P1	[2 3 1 4 5 3 2]
P2	[1 2 3 4 5 3 2]
P3	[4 1 2 5 3 3 2]

Terdapat 2 bagian kromosom, yaitu bagian 1 dan bagian 2, misal pada P1

						Bagian 2	
P1	2	3	1	4	5	3	2
	Bagian 1						

Bagian 1 merepresentasikan urutan path yang akan dilalui sales, Bagian 2 merepresentasikan pembagian titik kunjungan harian. Jadi untuk hari pertama terdapat 3 titik yang harus dikunjungi, yaitu 2, 3, dan 1. Sedangkan pada hari kedua terdapat 2 titik yang harus dikunjungi, yaitu 4 dan 5

4.3.3.2 Proses *Crossover*

Hasil dari inisialisasi pada proses sebelumnya bisa disebut sebagai *parent*, yang berarti terdapat 3 *parent* seperti yang terlihat pada Tabel 4.16. *Crossover* di sini adalah proses pertukaran sejumlah anggota *chromosome* antara dua *parent* yang dipilih secara acak. Pada proses ini akan dilakukan *crossover* dengan metode order crossover variasi pertama. Hasil dari proses *crossover* disebut sebagai *child*, di mana banyaknya *child* diperoleh dari hasil perkalian *crossover rate* dengan *popSize*. Kali ini *crossover rate* yang digunakan adalah 0,5 dan *popSize* 3, jika dikalikan akan diperoleh hasil 1,5 dan dibulatkan menjadi 2, sehingga terdapat 2 *child* yang akan dihasilkan dari proses *crossover* ini. Langkah pertama yang

dilakukan untuk *crossover* adalah dengan memilih dua *order point* (*op1* dan *op2*), di mana *order point* ini berfungsi sebagai batas atas dan batas bawah pertukaran *chromoshome*. *order point* di setiap generasi akan terus berubah dan dipilih secara acak bilangan antara 1 sampai 4. Pada iterasi ke-0 ini *order point* yang digunakan pada masing-masinga op1 dan op2 adalah 1 dan 2, kemudian pasangan *parent* yang terpilih adalah P1-P3.

Setelah *order point* dan *parent* dipilih, maka akan langsung menuju ke proses *crossover*. Untuk pasangan *parent* bisa dilihat pada Tabel 4.17 dan Tabel 4.18.

Tabel 4.17 *Chromosome* Pasangan *Parent* P1-P3

2	3	1	4	5
4	1	2	5	3

Seperti yang terlihat pada Tabel 4.17, kromosom yang digunakan pada proses *crossover* hanya pada bagian satu saja. Pada baris pertama adalah *chromosome* P1 yang akan menjadi *parent1* dan baris kedua adalah *chromosome* P3 yang akan menjadi *parent2*. Order point *op1* dan *op2* yang sebelumnya dipilih digunakan sebagai batas atas dan batas bawah, di mana element dari *parent* yang terletak diantara indeks *op1* dan *op2* akan tetap dipertahankan untuk *child*, sehingga akan terbentuk kerangka seperti berikut.

child 1	x	3	1	x	x

child 2	x	1	2	x	x

Kemudian untuk mengisi bagian kosong *child1*, salin semua *parent2* dimulai dari indeks ke *op2*, sehingga diperoleh hasil sebagai berikut:

Parent 2	4	1	2	5	3

hasil	5	3	4	1	2

Hilangkan element yang sudah ada di *child1*, sehingga diperoleh hasil sebagai berikut:

5	4	2

Hal yang sama juga dilakukan untuk mengisi bagian kosong *child2*, salin semua *parent1* dimulai dari indeks ke *op2*, sehingga diperoleh hasil sebagai berikut:

Parent 1	2	3	1	4	5

hasil	4	5	2	3	1

Hilangkan juga element yang sudah ada di *child2*, sehingga diperoleh hasil sebagai berikut:

4	5	3

kemudian sisipkan hasil ke setiap bagian *child* yang kosong dimulai setelah indeks ke *op2*, selanjutnyam masing-masing hasil *child1* dan *child2* digabungkan dengan kromosom bagian dua. Untuk hasilnya dapat dilihat pada Tabel 4.18.

Tabel 4.18 *Child* Hasil Proses *Crossover*

2	3	1	5	4	3	2
3	1	2	4	5	3	2

4.3.3.3 Proses Mutasi

Pada tahap ini akan dilakukan mutasi dengan menggunakan metode Simple Inverse Mutation. Mutasi merupakan proses pertukaran antar anggota *chromosome* dalam satu *parent.* Hasil dari proses mutasi disebut sebagai *child*, di mana banyaknya *child* diperoleh dari hasil perkalian *mutation rate* dengan *popSize*. Kali ini *mutation rate* yang digunakan adalah 0,5 dan *popSize* 3, jika dikalikan akan diperoleh hasil 1,5 dan dibulatkan menjadi 2, sehingga terdapat 2 *child* yang akan dihasilkan dari proses mutasi ini. Hal pertama yang dilakukan dalam mutasi adalah memilih *parent* secara random kemudian memilih 2 *reverse point (rp),* di mana *reverse point (rp)* merupakan batas atas dan batas bawah yang akan dibalik urutan posisinya. Untuk lebih jelas mengenai proses mutasi bisa dilihat pada Tabel 4.19.

Tabel 4.19 Proses Mutasi

	rp1		*rp2*	
1	2	3	4	5

Seperti yang dilihat pada Tabel 4.19, kromosom yang digunakan pada proses mutasi hanya pada bagian satu saja. Terdapat 2 *reverse point,* yaitu *RP1* dan *RP2*. Elemen mulai dari indeks ke *rp1* sampai *rp2* akan dibalik urutan posisinya. Hasil dari proses mutasi ini adalah 2 *child* dan masing-masing child digabungkan Kembali dengan kromosom bagian dua. Untuk hasilnya dapat dilihat pada Tabel 4.20.

Tabel 4.20 *Child* Hasil Mutasi

1	4	3	2	5	3	2
4	1	3	5	2	3	2

4.3.3.4 Evaluasi *Fitness*

Pada Tahap Evaluasi Fitnes akan menghitung nilai *fitness* setiap *chromosome*. Hasil dari proses *crossover* dan mutasi (*child*) akan digabungkan

menjadi satu dengan populasi awal kemudian baru dihitung nilai *fitnessnya* menggunakan Persamaan 4.1, di mana total jarak diperoleh dari penjumlahan jarak tempuh perjalanan sesuai dengan urutan yang terdapat dalam *chromosome* berdasarkan dari *distance matrix* yang telah dibuat sebelumnya. Proses perhitungan *fitness* ini dilakukan pada setiap klaster. Berikut adalah contoh perhitungan *fitness* pada klaster 3 dengan *chromosome* dan distance matrix dapat dilihat pada Tabel 4.21 dan Tabel 4.22.

Tabel 4.21 Gabungan *Parent* dan *Child*

No	Chromosome
P1	[2 3 1 4 5 3 2]
P2	[1 2 3 4 5 3 2]
P3	[4 1 2 5 3 3 2]
C1	[2 3 1 5 4 3 2]
C2	[3 1 2 4 5 3 2]
C3	[1 4 3 2 5 3 2]
C4	[4 1 3 5 2 3 2]

Contoh perhitungan *fitness* pada P1 dengan kromosom [2 3 1 4 5 3 2] dan *distance matrix* berdasarkan Tabel 4.11 *Distance Matrix* Klaster 3. Sesuai pembagian yang dilakukan kromosom bagian dua, hari pertama titik yang harus dikunjungi adalah [2 3 1] dan untuk hari kedua adalah [4 5]. Masing-masing urutan tersebut ditambahkan nilai 0 pada awal dan akhir kromosom, karena indeks 0 merupakan titik awal dan akhir kunjungan, yaitu Gudang PT Indomarco. sehingga perjalanan dimulai dari 0 kemudian secara berurutan akan mengunjungi 2,3,1 dan kemudian akan Kembali ke 0 lagi untuk hari pertama. Dan untuk hari kedua dimulai dari 0 kemudian 4, 5 dan kemudian akan Kembali ke 0 lagi. Untuk melihat jarak misal dari 2 ke 3, karena indeks dimulai dari 0, maka pada *distance matrix* dilihat pada baris ke 3 kolom ke 4 yang berarti jarak dari 2 ke 3 adalah 6,474292. Hal yang sama juga dilakukan terhadap urutan berikutnya. Untuk lebih jelasnya perhitungan *fitness* pada P1 menggunakan Persamaan 4.1 adalah sebagai berikut.

$$\text{Jarak P1 Hari 1} = 10{,}571861 + 6{,}474292 + 1{,}806649 + 8{,}180956 = 27{,}033758$$

$$\text{Jarak P1 Hari 2} = 11{,}800386 + 1{,}000138 + 10{,}881249 = 23{,}681772$$

Kemudian *fitness* P1 hari 1 dan hari 2 dihitung nilai fitnessnya.

$$Fitness\ P1 = \frac{100}{27{,}033758 + 23{,}681772} = 1{,}971782$$

Proses tersebut juga dilakukan terhadap P2, P3, C1, C2, C3, dan C4. Hasil dari perhitungan *fitness* dapat dilihat pada Tabel 4.22.

Tabel 4.22 Hasil perhitungan *Fitness*

No	Chromosome	jarak hari 1	jarak hari 2	total	fitnes
P1	[2 3 1 4 5 3 2]	27,03375851	23,68177244	50,71553	1,971783

P2	[1 2 3 4 5 3 2]	29,56805779	23,68177244	53,24983	1,87794
P3	[4 1 2 5 3 3 2]	34,29205606	27,2274352	61,51949	1,625501
C1	[2 3 1 5 4 3 2]	27,03375851	23,68177244	50,71553	1,971783
C2	[3 1 2 4 5 3 2]	27,29131982	23,68177244	50,97309	1,961819
C3	[1 4 3 2 5 3 2]	31,94537651	21,80269705	53,74807	1,860532
C4	[4 1 3 5 2 3 2]	29,77881293	21,80269705	51,58151	1,938679

4.3.3.5 Proses Seleksi

Metode yang digunakan dalam proses seleksi adalah *elitism selection,* yaitu dengan melakukan pengurutan nilai *fitness* dari yang terbesar ke terkecil. Setelah diurutkan, akan dipilih dindividu sebanyak *popsize* yang diambil mulai dari urutuan teratas untuk generasi selanjutnya. Hasil pengurutan dan seleksi bisa dilihat pada Tabel 4.23 dan 4.24.

Tabel 4.23 Hasil Pengurutan

No	Chromosome	jarak hari 1	jarak hari 2	total	fitnes
P1	[2 3 1 4 5 3 2]	27,033758	23,681772	50,715530	1,971783
C1	[2 3 1 5 4 3 2]	27,033758	23,681772	50,715530	1,971783
C2	[3 1 2 4 5 3 2]	27,291319	23,681772	50,973090	1,961819
C4	[4 1 3 5 2 3 2]	29,778812	21,802697	51,581510	1,938679
P2	[1 2 3 4 5 3 2]	29,568057	23,681772	53,249830	1,877940
C3	[1 4 3 2 5 3 2]	31,945376	21,802697	53,748070	1,860532
P3	[4 1 2 5 3 3 2]	34,292056	27,227435	61,519490	1,625501

Tabel 4.24 Individu terpilih

No	Chromosome	jarak hari 1	jarak hari 2	total	fitnes
P1	[2 3 1 4 5 3 2]	27,03375851	23,68177244	50,71553	1,971783
C1	[2 3 1 5 4 3 2]	27,03375851	23,68177244	50,71553	1,971783
C2	[3 1 2 4 5 3 2]	27,29131982	23,68177244	50,97309	1,961819

Rangkaian proses di atas mulai inisialisasi *chromosome*, *crossover*, mutasi, evaluasi, hingga seleksi merupakan rangkaian proses pada iterasi pertama. Istilah iterasi pada Algoritma Genetika biasa disebut generasi, yang bisa diartikan juga bahwa rangkaian proses di atas merupakan generasi pertama. Untuk generasi kedua, individu-individu yang digunakan adalah individu terpilih dari generasi pertama yang dapat dilihat pada Tabel 4.24. Untuk rangkaian proses pada generasi kedua juga sama seperti generasi pertama. Dimulai dari Inisialisasi *chromosome* yang hasilnya bisa dilihat pada Tabel 4.25.

Tabel 4.25 Inisialisasi *Chromosome* Generasi Kedua

No	Chromosome
P1	[2 3 1 4 5 3 2]

P2	[2 3 1 5 4 3 2]
P3	[3 1 2 4 5 3 2]

Langkah selanjutnya adalah melakukan *crossover* dari 2 *parent* yang dipilih secara acak. *Order point* op1 dan op2 yang digunakan adalah 2 dan 3, kemudian *child* yang akan dihasilkan sebanyak 2 (dari hasil perkalian *crossover rate* dan *popSize*), sehingga terdapat 2 pasang *parent* yang akan mengalami *crossover.* Hasil dari *crossover* dapat dilihat pada Tabel 4.26.

Tabel 4.26 Hasil *Crossover* Generasi Kedua

3	1	2	4	5	3	2
3	2	1	4	5	3	2

Kemudian langkah selanjutnya adalah mutasi yang akan menghasilkan *child* sebanyak 2 (dari hasil perkalian *mutation rate* dan *popSize*), sehingga terdapat 2 *parent* yang akan mengalami mutasi. Hasil dari mutasi dapat dilihat pada Tabel 4.27.

Tabel 4.27 Hasil Mutasi Generasi Kedua

2	3	5	1	4	3	2
2	3	5	4	1	3	2

Setelah proses *crossover* dan mutasi selesai akan dilakukan evaluasi *fitness*. Semua individu baik *parent* maupun *child* akan dihitung *fitness* dengan rumus Persamaan 4.1. Hasil dari proses evaluasi *fitness* dapat dilihat pada Tabel 4.28.

Tabel 4.28 Hasil Evaluasi *Fitness* Generasi Kedua

No	Chromosome	jarak hari 1	jarak hari 2	total	fitnes
P1	[2 3 1 4 5 3 2]	27,033758	23,681772	50,71553	1,971783
P2	[2 3 1 5 4 3 2]	27,033758	23,681772	50,71553	1,971783
P3	[3 1 2 4 5 3 2]	27,291319	23,681772	50,97309	1,961819
C1	[3 1 2 4 5 3 2]	27,291319	23,681772	50,97309	1,961819
C2	[3 2 1 4 5 3 2]	29,568057	23,681772	53,24983	1,877940
C3	[2 3 5 1 4 3 2]	34,691199	26,570730	61,26193	1,632335
C4	[2 3 5 4 1 3 2]	34,691199	26,570730	61,26193	1,632335

Langkah setelah proses Evaluasi *Fitness* selesai adalah melakukan seleksi dengan cara mengurutkan hasil evaluasi *fitness* dari yang terbesar ke terkecil. Setelah itu akan dipilih 3 individu teratas (sesuai dengan jumlah *popSize*). Hasil dari pengurutan dan seleksi dapat dilihat pada Tabel 4.29 dan Tabel 4.30.

Tabel 4.29 Hasil Pengurutan Generasi Kedua

No	Chromosome	jarak hari 1	jarak hari 2	total	fitnes

P1	[2 3 1 4 5 3 2]	27,033758	23,681772	50,715530	1,971783
P2	[2 3 1 5 4 3 2]	27,033758	23,681772	50,715530	1,971783
P3	[3 1 2 4 5 3 2]	27,291319	23,681772	50,973090	1,961819
C1	[3 1 2 4 5 3 2]	27,291319	23,681772	50,973090	1,961819
C2	[3 2 1 4 5 3 2]	29,568057	23,681772	53,249830	1,877940
C3	[2 3 5 1 4 3 2]	34,691199	26,570730	61,261930	1,632335
C4	[2 3 5 4 1 3 2]	34,691199	26,570730	61,261930	1,632335

Tabel 4.30 Individu Terpilih Generasi Kedua

No	Chromosome	jarak hari 1	jarak hari 2	total	fitnes
P1	[2 3 1 4 5 3 2]	27,033758	23,681772	50,715530	1,971783
P2	[2 3 1 5 4 3 2]	27,033758	23,681772	50,715530	1,971783
P3	[3 1 2 4 5 3 2]	27,291319	23,681772	50,973090	1,961819

Karena jumlah generasi (*jmlGen*) yang diinisialisasi di awal adalah sebanyak 2, maka pada generasi kedua ini iterasi dihentikan. Sehingga pada klaster ketiga individu yang merupakan solusi adalah Individu terpilih terakhir yang memiliki *fitness* tertinggi, yaitu pada P1 dengan jarak 50,715530 km. Untuk mencari rute optimal pada klaster 1 dan 2 rangkaian prosesnya juga sama seperti di atas.

4.4 Perancangan Pengujian

Pada tahap ini akan dilakukan pengujian terhadap Algoritma Genetika yang bertujuan untuk mengetahui parameter terbaik yang paling optimal pada kasus distribusi Produk PT Indomarco Adi Prima *Stock Point Nganjuk*. Parameter yang digunakan dalam pengujian adalah sebagai berikut.

1. Pengujian kualitas klaster
2. Ukuran populasi
3. Banyak Generasi
4. Kombinasi *crossover rate* (*cr*) dan *mutation rate* (*mr*)
5. Perbandingan hasil dari gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan Algoritma Genetika tanpa *K-Means*
6. Perbandingan hasil dari gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan Rute yang Digunakan Perusahaan

Dalam perancangan pengujian ini, proses pengujian dilakukan secara berulang untuk bisa membandingkan mana nilai parameter terbaik. Untuk pengujian poin 1 sampai 3 Setiap parameter akan dilakukan pengujian sebanyak 10 kali, karena karena dengan perulangan sebanyak itu sudah bisa menghasilkan solusi yang beragam. Pada setiap iterasi akan didapatkan nilai *fitness* dari setiap parameter yang nantinya akan digunakan sebagai solusi dalam pencarian rute

optimal ini. Sedangkan untuk pengujian pada point 4 dan 5 hasil dari metode GKA akan diambilkan berdasarkan hasil terbaik dari pengujian sebelumnya.

4.4.1 Pengujian Kualitas Klaster

Pada pengujian kualitas klaster ini dilakukan perbandingan kualitas antara klaster hasil yang diperoleh dari K-Means dengan klaster dari data riil menggunakan metode *Silhouette Coefficient.* Pengujian dilakukan dengan membandingkan nilai *Silhouette Coefficient* dari hasil *K-Means* dengan nilai *Silhouette Coefficient* dari klaster data rute yang digunakan perusahaan saat ini. Setiap pengujian dilakukan sebanyak 3 kali, nilai *Silhouette Coefficient* yang mendekati 1 merupakan klaster dengan kualitas baik. Perancangan pengujian terhadap kualitas klaster dapat dilihat pada Tabel 4.31.

Tabel 4.31 Perancangan Pengujian Kualitas Klaster

NO	Jumlah Data	Banyak Klaster	Nilai *Silhouette Coefficient*	
			Hasil *K-Means*	Data asli tanpa K-Means
1				
2				
3				

4.4.2 Pengujian Ukuran Populasi

Pengujian ini dilakukan untuk mengetahui nilai *fitness* yang paling optimal serta melihat pengaruh banyaknya *popSize* yang digunakan terhadap nilai *fitness* yang dihasilkan. Pada penelitian ini banyak *popSize* yang digunakan untuk pengujian adalah kelipatan 200 dimulai dari 200 sampai 2000. Setiap banyak *popSize* akan dilakukan pengujian sebanyak 10 kali percobaan dan kemudian dihitung rata-rata *fitness*nya. Jumlah *popSize* yang menghasilkan rata-rata nilai *fitness terbesar* akan digunakan sebagai solusi akhir. Perancangan pengujian terhadap banyaknya *popSize* dapat dilihat pada Tabel 4.32.

Tabel 4.32 Perancangan Pengujian Terhadap Banyaknya *popSize*

Banyak *PopSize*	*Fitness* Percobaan ke-										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
200											
400											
600											
800											
1000											
1200											
1400											
1600											

1800										
2000										

4.4.3 Pengujian Jumlah Generasi

Pengujian jumlah generasi bertujuan untuk mengetahui pengaruh jumlah generasi terhadap nilai *fitness* yang dihasilkan. Selain itu dengan melakukan pengujian jumlah generasi ini akan dapat diketahui jumlah iterasi optimal yang digunakan untuk mencari solusi. Pada penelitian ini jumlah generasi yang digunakan untuk pengujian adalah kelipatan 100 dimulai dari 200 sampai 1000. Pada setiap generasi akan dilakukan pengujian sebanyak 10 kali percobaan dan kemudian dihitung rata-rata *fitness* dari 10 kali percobaan tersebut. Banyak Generasi yang menghasilkan rata-rata nilai *fitness terbesar* akan digunakan sebagai solusi akhir. Perancangan pengujian terhadap jumlah generasi dapat dilihat pada Tabel 4.33.

Tabel 4.33 Perancangan Pengujian Terhadap Jumlah Generasi

Banyak Generasi	*Fitness* Percobaan ke-										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
200											
300											
400											
500											
600											
700											
800											
900											
1000											

4.4.4 Pengujian Kombinasi *Crossover rate* dan *Mutation rate*

Pengujian kombinasi *crossover rate (cr)* dan *mutation rate (mr)* bertujuan untuk mengetahui pengaruh *cr* dan *mr* terhadap nilai *fitness* yang dihasilkan. Kombinasi *cr* dan *mr* yang digunakan pada penelitian ini adalah 0,1:0,9; 0,2:0,8; 0,3:0,7; 0.4:0,6; 0,5:0,5; 0,6:0,4; 0,7:0,3; 0,8:0,2; 0,9:0,1, di mana setiap kombinasi *cr* dan *mr* akan dilakukan pengujian sebanyak 10 kali percobaan dan kemudian dihitung rata-rata *fitness* dari 10 kali percobaan tersebut. kombinasi *cr* dan *mr* yang menghasilkan rata-rata nilai *fitness terbesar* akan digunakan sebagai solusi

akhir. Perancangan pengujian terhadap kombinasi *cr* dan *mr* dapat dilihat pada Tabel 4.34.

Tabel 4.34 Perancangan Pengujian Kombinasi *cr* dan *mr*

Crossover rate: *Mutation rate*	*Fitness* Percobaan ke-										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
0,1:0,9											
0,2:0,8											
0,3:0,7											
0,4:0,6											
0,5:0,5											
0,6:0,4											
0,7:0,3											
0,8:0,2											
0,9:0,1											

4.4.5 Pengujian Perbandingan gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan Algoritma Genetika tanpa *K-Means*

Pengujian ini bertujuan untuk membandingkan solusi yang diperoleh dari gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan Algoritma Genetika tanpa *K-Means.* Pada pengujian ini parameter yang digunakan untuk membandingkan keduanya adalah sama, yaitu hasil terbaik yang telah diperoleh dari keseluruhan pengujian sebelumnya. Perancangan pengujian perbandingan gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan Algoritma Genetika tanpa *K-Means* dapat dilihat pada Tabel 4.35.

Tabel 4.35 Perancangan Pengujian Perbandingan gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan Algoritma Genetika tanpa *K-Means*

Algoritma	Gabungan K-Means dan GA	GA Tanpa K-Means
Jumlah titik kunjungan		
Jumlah *Sales*		
Jumlah populasi		
Jumlah generasi		
Total Jarak (km)		
Fitness		

4.4.6 Pengujian Perbandingan gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan Rute yang Digunakan Perusahaan

Pengujian ini bertujuan untuk membandingkan solusi yang diperoleh dari gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan rute asli yang saat ini digunakan oleh PT Indomarco Adi Prima (*Stock Point Nganjuk*). Pada pengujian ini yang akan dibandingkan adalah total jarak tempuh dari keduanya, di mana GKA akan menggunakan hasil terbaik dari pengujian yang sebelumnya dilakukan, sedangkan rute asli akan menghitung total jarak tempuh dari rencana perjalanan *sales* yang ada. Selain itu juga akan dihitung selisih jarak tempuh antara hasil yang diperoleh dari GKA dengan rute asli yang digunakan saat ini.

BAB 5 IMPLEMENTASI

Pada bab implementasi ini akan dilakukan penerapan sesuai dengan perancangan yang telah dibuat pada bab sebelumnya. Pada implementasi ini proses penerapan akan dibagi ke dalam beberapa bagian, yaitu implementasi Algoritma *K-Means*, implementasi menghitung distance matrix, dan implementasi Algoritma Genetika.

5.1 Implementasi Algoritma *K-Means*

Pada bagian implementasi Algoritma *K-Means* ini akan berisi penjelasan kode program dalam menjalankan Algoritma *K-Means* secara umum mulai dari inisialisasi awal beberapa atribut seperti jumlah klaster, hingga hasil klaster diperoleh. Di sini tujuan dari Algoritma *K-Means* adalah untuk membagi titik-titik wilayah yang akan dikunjungi menjadi beberapa kelompok sesuai jumlah kendaraan(*sales*) yang ada.

5.1.1 Implementasi Inisialisasi *Centroid* Awal

Pada bagian ini akan dilakukan inisialisasi *centroid* awal dengen menggunakan metode *Binary Search Centroid*. Kode program untuk implementasi *generate centroid* awal ini terdapat pada Kode Program 5.1.

```
    Generate Centroid
1   def generate_centroid(dt_latlong,k):
2       c = np.empty((0, 2), int)
3
4       lat_max = max(dt_latlong[:, 0])
5       lat_min = min(dt_latlong[:, 0])
6       long_max = max(dt_latlong[:, 1])
7       long_min = min(dt_latlong[:, 1])
8
9       M = np.asarray([(lat_max-lat_min)/3, (long_max-long_min)/3])
10
11      for i in range(1,k+1):
12          c=np.append(c,[[((lat_min)+((i-1)*M[0])), (long_min+((i-
            1)*M[1]))]], axis=0)
13
14      return c
```

Kode Program 5.1 Inisialisasi *Centroid Awal*

Pembahasan untuk implementasi Kode Program 5.1 akan dijabarkan sebagai berikut:

1. Baris 1 mendefinisikan fungsi Bernama *generate_centroid* yang di dalamnya nanti terdapat rangkaian proses untuk menghitung *centroid* awal dengan menggunakan metode *binary search.*
2. Baris 2 inisialisasi variabel *c* yang akan menyimpan hasil *centroid* awal.
3. Baris 4-7 mencari nilai minimum dan maksimum dari setiap atribut *latitude* dan *longitude* yang disimpan pada variablel *lat_max, lat_min, long_max, long_min.*

4. Baris 9 melakukan perhitungan jarak antar *centroid* awal yang akan dibentuk yang disimpan dalam variabel *M*.
5. Baris 11-12 melakukan perhitungan untuk menentukan *centroid* awal berdasarkan nilai minimal-maksimal setiap atribut dan *M* yang merupakan jarak spesifik antar *centroid*.
6. Baris 14 mengembalikan nilai c yang merupakan *centroid* awal.

5.1.2 Implementasi *Euclidian Distance*

Pada bagian ini merupakan penerapan dari rumus *euclidian distance* berdasarkan Persamaan 2.1 yang akan digunakan untuk menghitung jarak antara data terhadap setiap *centroid*. Kode program untuk implementasi generate *centroid* awal ini terdapat pada Kode Program 5.2.

Euclidian Distance

```
1  def euclidiandist(data, c):
2      temp = abs(data-c)
3      kuadrat = np.dot(temp.T, temp)
4      euclid = np.sqrt(kuadrat)
5      return euclid
6
```

Kode Program 5.2 Implementasi *Euclidian Distance*

Pembahasan untuk implementasi Kode Program 5.2 akan dijabarkan sebagai berikut:

1. Baris 1 mendefinisikan fungsi dengan nama *euclidian distance* yang di dalamnya terdapat rangkaian proses perhitungan jarak.
2. Baris 2-3 merupakan proses perhitungan jarak menggunakan *euclidian* yang hasil perhitungannya disimpan dalam variabel *euclid.*
3. Melakukan *return value* berupa variabel *euclid*

5.1.3 Impementasi *K-Means*

Pada bagaian implementasi *K-Means* ini merupakan penjelasan kode program dalam menjalankan algoritma *K-Means* secara keseluruhan meliputi inisialisasi *centroid* awal, menghitung jarak dengan *euclidian distance*, hingga menyimpan hasil klasterisasi ke dalam file berformat excel. Kode program untuk implementasi generate *centroid* awal ini terdapat pada Kode Program 5.3.

K-Means

```
1  def kmeans(data):
2      k = 3
3      h, w = np.shape(data)
4      cluster = np.zeros(h)
5      count_cluster = np.zeros(k)
6      iterasi = 0
7      stop = False
8
9      centroid = generate_centroid(data[:, 4:])
10
11     while stop == False:
12         c1 = np.empty((0, 6), int)
```

```
        c2 = np.empty((0, 6), int)
        c3 = np.empty((0, 6), int)
        lbl = []

        for i in range(len(data[:, 4:])):
            x = 1000
            counter = 0

           for j in range(len(centroid)):
                counter += 1
                euclid = euclidiandist(data[i, 4:], centroid[j])
                if euclid < x:
                    cluster[i] = counter
                    x = euclid

            for s in range(len(count_cluster)):
             count_cluster[s] = np.count_nonzero(cluster == s+1)

            if cluster[i] == 1:
                c1 = np.append(c1, np.array([data[i]]), axis=0)
                lbl.append(1)
            elif cluster[i] == 2:
                c2 = np.append(c2, np.array([data[i]]), axis=0)
                lbl.append(2)
            elif cluster[i] == 3:
                c3 = np.append(c3, np.array([data[i]]), axis=0)
                lbl.append(3)

        newcentroid = np.array([
                     np.sum(c1[:, 4:],axis=0)/count_cluster[0],
                     np.sum(c2[:, 4:],axis=0)/count_cluster[1],
                     np.sum(c3[:, 4:],axis=0)/count_cluster[2]])

        cek = abs(np.subtract(centroid, newcentroid))

        if np.sum(cek) == 0:
            stop = True
        else:
            iterasi += 1
            centroid = newcentroid
            stop = False

    X = data[:, 4:]
    Y = lbl
    print("Silhouette score =", silhouette_score(X, Y,
    metric='euclidean'))


    c1 = np.insert(c1, 0, [0, "Start Point", "PT Indomarco Adi
    Prima","Jl lenjen suprapto no 70", -7.616966,
    111.894101],axis=0)
    c2 = np.insert(c2, 0, [0, "Start Point", "PT Indomarco Adi
    Prima","Jl lenjen suprapto no 70", -7.616966,
    111.894101],axis=0)
    c3 = np.insert(c3, 0, [0, "Start Point", "PT Indomarco Adi
    Prima","Jl lenjen suprapto no 70", -7.616966,
    111.894101],axis=0)

    return centroid, c1, c2, c3
```

Kode Program 5.3 Implementasi K-Means

Pembahasan untuk implementasi Kode Program 5.3 akan dijabarkan sebagai berikut:

1. Baris 1-7 melakukan inisialisasi terhadap beberapa attribute seperti k yang merupakan jumlah klaster yang akan dibuat pada Algoritma *K-Means* ini yang disesuaikan dengan jumlah *sales.* Kemudian juga menginisialisasi *array* yang bernama *cluster* yang nanti akan menyimpan klaster setiap data berupa angka 1 sampa k klaster. *count_cluster* yang akan menyimpan jumlah angota setiap klaster. *iterasi* untuk menghitung banyaknya iterasi, serta *stop* yang bertipe *boolean* untuk menyimpan status kondisi perulangan.
2. Baris 9 memaggil fungsi *generate_centroid* dengan parameter data *latitude* dan *longitude* untuk mendapatkan *centroid* awal.
3. Baris 11 merupakan perulangan dengan menggunakan *while* dengan kondisi *stop == false,* yang artinya ketika variabel *stop* masih bernilai *false,* maka perulangan akan terus berjalan
4. Baris 12-15 melakukan inisialisasi terhadap variable *c1, c2, c3, lbl* yang akan data menyimpan anggota masing-masing klaster 1, klaster 2, klaster 3, dan label untuk setiap data dari hasil klastering.
5. Baris 17 merupakan perulangan dengan menggunakan *for* sebanyak *data*.
6. Baris 18-19 melakukan inisialisasi terhadap variabel *x* yang akan menyimpan data dengan nilai jarak euclidian terkecil setiap data. Variable *counter* digunakan untuk menyimpan nilai *cluster* sementara.
7. Baris 21-26 merupakan perulangan untuk menghitung jarak dengan euclidian setiap data terhadap *centroid*, yang sekaligus menentukan setiap data tersebut masuk ke dalam klaster berapa.
8. Baris 28-29 melakukan penghitungan total anggota setiap klaster.
9. Baris 31-39 akan menyimpan hasil klastering, di mana anggota setiap klaster akan dipisah dan disimpan pada masing-masing variabel *c1, c2,* dan *c3.* Dan akan menyimpan label sesuai hasil dari klastering berupa angka 1, 2, atau 3.
10. Baris 41--43 akan menghitung *centroid* baru berdasarkan hasil klastering sebelumnya yang akan disimpan ke dalam variabel *newcentroid*.
11. Baris 45-52 melakukan operasi pengurangan setiap elemen dari variabel *c* terhadap *newcentroid*. Tujuannya adalah untuk melakukan pengecekan apakah masih ada sisa (tidak nol) dari operasi pengurangan tersebut, jika masih ada sisa berarti masih ada perubahan terhadap anggota setiap klaster baik perubahan dari segi jumlah maupun pergantian klaster anggota. Itu artinya perulangan akan terus berjalan dan Kembali lagi ke baris 12 namun dengan *centroid* baru yang tadi disimpan di variabel *newcentroid,* serta variabel iterasi ditambah 1 pada setiap itersasi.
12. Baris 54-57 melakukan pengujian hasil klastering menggunakan silhouette score.

13. Baris 59-61 melakukan *insert* berupa data lokasi PT Indomarco Adi Prima (*Stock Point* Nganjuk) terhadap array setiap klaster pada indeks ke-0. *Insert* ini dilakukan karena data tersebut merupakan titik keberangkatan, di mana seluruh klaster memiliki titik keberangkatan yang sama.
14. Baris 63 mengembalikan nilai berupa *centroid* dan hasil klasterisasi.

5.2 Implementasi Membuat Distance Matrix

Pada bagian implementasi membuat *distance matrix* ini akan berisi penjelasan kode program mengenai penggunaan formula haversine untuk membuat *distance matrix*. Di sini fungsi dari formula haversine adalah untuk menghitung jarak antara dua titik lokasi melalui data input berupa data *latitude* dan *longitude*. Kode program untuk implementasi membuat *distance matrix* ini terdapat pada Kode Program 5.4.

Formula Haversine

```
1  from math import radians, cos, sin, asin, sqrt
2
3  def haversine(data_cluster):
4        m, n = data_cluster.shape
5        hasil = np.zeros((m, m))
6
7        for i in range(len(data_cluster)):
8            for j in range(len(data_cluster)):
9
10               lat1 = data_cluster[i,0]
11               lon1 = data_cluster[i,1]
12               lat2 = data_cluster[j,0]
13               lon2 = data_cluster[j,1]
14
15               R = 6371.1
16
17               dLat = radians(lat2 - lat1)
18               dLon = radians(lon2 - lon1)
19               lat1 = radians(lat1)
20               lat2 = radians(lat2)
21
22               a = sin(dLat/2)**2 + cos(lat1)*cos(lat2)*sin(dLon/2)**2
23               c = 2*asin(sqrt(a))
24               hasil[i, j] = R * c
25
26       return hasil
```

Kode Program 5.4 Formula Haversine

Pembahasan untuk implementasi Kode Program 5.4 akan dijabarkan sebagai berikut:

1. Baris 1 melakukan inisiali terhadap modul *math*, di mana dari modul *math* ini yang akan digunakan adalah *radians, cos, sin, asin, sqrt*.
2. Baris 3 melakukan deklarasi fungsi dengan nama haversin, di mana fungsi ini memiliki parameter berupa *data_cluster*, yaitu data setiap klaster.
3. Baris 4-5 melakukan deklarasi terhadap variabel *m* dan *n* yang digunakan untuk menyimpan bentuk panjang matrix masing-masing *m* dan *n* adalah

untuk baris dan kolom. Variabel hasil untuk membuat sebuah matriks yang nantinya akan menyimpan hasil dari pembuatan *distance matrix*.

4. Baris 7-8 merupakan sebuah deklarasi perulangan, di mana perulangan pada baris ke 7 digunakan untuk mengambil data *latitude* dan *longitude* titik asal, sedangkan baris 8 untuk mengambil data *latitude* dan *longitude* titik tujuan. Kedua perulangan tersebut dijalankan secara berurutan dan tidak dipisah.
5. Baris 10-13 melakukan deklarasi terhadap beberapa variable, seperti *lat1* dan *lon1* untuk menyimpan data *latitude* dan *longitude* titik asal, serta *lat2 dan lon2* untuk menyimpan data *latitude* dan *longitude* titik tujuan. Keempat variabel tersebut berjalan berdasarkan perulangan pada baris 7-8.
6. Baris 15 melakukan inisialisasi terhadap variabel *R*, di mana *R* akan menyimpan nilai radius bumi yang sudah ditetapkan. Karena ingin menggunakan jarak dengan satuan kilometer, maka radius yang digunakan adalah 6371,1.
7. Baris 17-20 melakukan inisialisasi terhadap beberapa variabel yang nanti akan digunakan dalam rumus haversine, seperti *dLat* yang menyimpan hasil pengurangan *latitude* tujuan dan *latitude* asal*, dLon* yang menyimpan hasil pengurangan *longitude* tujuan dan *longitude* asal*.*
8. Baris 22-24 melakukan perhitungan dengan formula haversine menggunakan Persamaan 2.5. Hasilnya akan disimpan pada variabel array hasil yang disesuaikan dengan indeksnya.
9. Baris 26 mengembalikan nilai berupa variable hasil.

5.2.2 Implementasi Algoritma Genetika

Pada bagian implementasi membuat Algoritma Genetika ini akan berisi penjelasan kode program mengenai penggunaan Algoritma Genetika mulai dari inisialisasi kromosom, *crossover*, hingga diperoleh hasil akhir berupa rute paling pendek dari setiap klaster.

5.2.3 Implementasi *Crossover*

Pada bagian implementasi *crossover* ini merupakan proses tukar silang antar *parent* terpilih yang akan menghasilkan *child* berdasarkan nilai *crossover rate* (*cr*). Kode program untuk implementasi *crossover* ini terdapat pada Kode Program 5.5.

	Algoritma Genetika : *Crossover*
1	`def crossover(pop, jml_child,k):`
2	`    child = np.empty((0, len(pop[0])), int)`
3	`    for i in range(jml_child):`
4	`        offspring1 = []`
5	`        offspring2 = []`
6	`        xa = []`
7	`        xb = []`
8	
9	`        op1 = np.random.randint(5, len(pop[0]-k)-5)`
10	`        op2 = np.random.randint(5, len(pop[0]-k)-5)`
11	

```
12          while op2 == op1:
13              op2 = np.random.randint(5, len(pop[0]-k)-5)
14
15          if op2 < op1:
16              op1, op2 = op2, op1
17
18          p1 = np.random.randint(0, len(pop))
19          p2 = np.random.randint(0, len(pop))
20
21          while p1 == p2:
22              p2 = np.random.randint(0, len(pop))
23
24          parent1 = pop[p1]
25          parent2 = pop[p2]
26
27          temp1 = parent2[op2+1:len(parent2)-k]
28          temp1 = np.append(temp1, parent2[:op2+1])
29
30          temp2 = parent1[op2+1:len(parent1)-k]
31          temp2 = np.append(temp2, parent1[:op2+1])
32
33          offspring1 = np.array(parent1[op1:op2+1])
34          offspring2 = np.array(parent2[op1:op2+1])
35
36          mask1 = ~np.isin(temp1, offspring1)
37          xa = temp1[mask1]
38          mask2 = ~np.isin(temp2, offspring2)
39          xb = temp2[mask2]
40
41          part2 = parent1[len(parent1)-k:]
42
43          offspring1 = np.append(offspring1,
            xa[:len(parent1[op2+1:])-k])
44          offspring1 = np.insert(offspring1, 0,
            xa[len(parent2[op2+1:])-k:])
45          offspring1 = np.append(offspring1, part2,axis = 0)
46
47          offspring2 = np.append(offspring2,
            xb[:len(parent1[op2+1:])-k])
48          offspring2 = np.insert(offspring2, 0,
            xb[len(parent2[op2+1:])-k:])
49          offspring2 = np.append(offspring2, part2,axis = 0)
50
51          if len(child) < jml_child:
52              child = np.append(child, [offspring1], axis=0)
53              if len(child) < jml_child:
54                  child = np.append(child, [offspring2], axis=0)
55          else:
56              break
57      return child
```

Kode Program 5.5 Implementasi *Crossover*

Pembahasan untuk implementasi Kode Program 5.5 akan dijabarkan sebagai berikut:

1. Baris 1 merupakan deklarasi fungsi dengan nama *crossover*, di mana di dalam fungsi ini nanti akan dilakukan proses *crossover* dengan metode order *crossover*.
2. Baris 2 melakukan inisialisasi variabel array bernama *child*. Variabel tersebut nanti akan digunakan untuk menyimpan hasil *crossover*.
3. Baris 3 melakukan inisialisasi perulangan sejumlah *child* yang akan dibuat.

4. Baris 4-7 melakukan inisialisasi terhadap beberapa variabel array, diantaranya adalah *offspring1* dan *offspring2* untuk menyimpan hasil *child* yang dibentuk, serta *xa* dan *xb* untuk menyimpan data untuk keperluan proses *crossover*.
5. Baris 9-16 melakukan inisialisasi terhadap variabel *op1* dan *op2* yang akan digunakan untuk menyimpan nilai operator untuk batas atas dan batas bawah proses *crossover*. Variabel *op1* dan *op2* harus berbeda dan *op1* harus lebih kecil dari *op2*.
6. Baris 18-25 melakukan inisialisasi terhadap variabel *p1* dan *p2*, di mana variabel tersebut digunakan untuk menyimpan nilai indeks. Variabel *parent1* dan *parent2* akan memilih data dengan indeks *p1* dan *p2* pada dataset untuk dijadikan sebagai *parent*.
7. Baris 27-31 merupakan proses inti dari proses *crossover*. Variabel *temp1* untuk menyimpan sementara element dari *parent2* dimulai dari indeks *op2* sampai indeks terakhir kemudian dilanjutkan element indeks pertama sampai element sebelum indeks ke *op2*. Begitu juga variabel *temp2* untuk menyimpan data dari *parent1*.
8. Baris 33 dan 34 menyimpan element dari setiap *parent* pada indeks *op1* sampai *op2*, masing-masing disimpan pada variabel *offspring1* dan *offspring2*.
9. Baris 36-39 melakukan pengecekan terhadap setiap element dari variabel *temp1* dan *temp2* apakah sama dengan element pada *offspring1* maupun *offspring2*, jika tidak sama maka element tersebut disimpan pada variabel *xa* dan *xb.*
10. Baris 41 melakukan inisialisasi *part2* untuk menyimpah kromosom bagian 2 yang merepresentasikan pembagian banyak rute untuk 5 hari kerja.
11. Baris 43-49 menggabungkan *xa* dengan *offspring1* dan *xb* dengan *offspring2* untuk membentuk *child* baru. Kemudian menggabungkannya dengan *part2*.
12. Baris 51-56 menyimpan setiap *child* baru pada variabel *child*, sekaligus melakukan pengecekan apakah jumlah *child* sudah terpenuhi atau belum, jika belum maka perulangan untuk membentuk *child* baru terus dilakukan, jika tidak maka perulangan akan berhenti.
13. Baris 57 mengembalikan nilai berupa variabel *child* yang menyimpan hasil proses *crossover*.

5.2.4 Implementasi Mutasi

Pada bagian implementasi mutasi ini merupakan proses tukar silang antar kromosom pada satu *parent* terpilih terpilih yang akan menghasilkan *child* berdasarkan nilai *mutation rate* (*mr*). Kode program untuk implementasi mutasi ini terdapat pada Kode Program 5.6.

	Algoritma Genetika : Mutasi
59	def mutasi(pop, jml_child,k):

```
60      child = np.empty((0, len(pop[0])), int)
61
62      for i in range(jml_child):
63
64          indeks = np.random.randint(0, len(pop))
65          parent = np.array(pop[indeks])
66
67          rp1 = np.random.randint(0, len(parent)-k)
68          rp2 = np.random.randint(0, len(parent)-k)
69
70          while rp2 == rp1:
71              rp2 = np.random.randint(0, len(parent)-k)
72
73          if rp2 < rp1:
74              rp1, rp2 = rp2, rp1
75
76          parent[rp1:rp2+1] = np.flip(parent[rp1:rp2+1])
77
78          child = np.append(child, [parent], axis=0)
79
80      return child
```

Kode Program 5.6 Implementasi Proses Mutasi

Pembahasan untuk implementasi Kode Program 5.6 akan dijabarkan sebagai berikut:

1. Baris 59 merupakan deklarasi fungsi dengan nama mutasi, di mana di dalam fungsi ini nanti akan dilakukan proses mutasi dengan metode *inverse mutation*.
2. Baris 60 melakukan deklarasi variabel dengan nama *child* yang akan digunakan untuk menyimpan hasil mutasi.
3. Baris 62 melakukan inisialisasi perulangan sebanyak *child* yang akan dibentuk.
4. Baris 64 dan 65 memilih *parent* secara acak yang akan dilakukan mutasi
5. Baris 67 dan 74 memilih dua *reverse point* secara acak yang akan digunakan sebagai batas atas dan batas bawah element *parent* yang akan dibalik urutan posisinya. Kedua *reverse point* nilainya harus berbeda.
6. Baris 76-78 merupakan proses inti mutasi, yaitu posisi elemen mulai dari *rp1* sampai *rp2* pada parent terpilih dibalik urutan posisinya. Kemudian hasil dari mutasi akan disimpan dalam variabel *child*.
7. Baris 80 mengembalikan nilai berupa variabel *child* yang berisi hasil dari proses mutasi.

5.2.5 Implementasi Hitung *Fitness*

Pada bagian implementasi hitung *fitness* ini merupakan proses perhitungan yang menghasilkan nilai *fitness* pada setiap kromosom. Kode program untuk implementasi hitung *fitness* ini terdapat pada Kode Program 5.7.

```
    Algoritma Genetika : Hitung Fitness
82  def Hitung_fitness(allpop, dmatrix,k):
83      jarak = np.empty((0,k),float)
```

```
84      fitness = np.empty((0,1),float)
85
86      for j in range(len(allpop)):
87          paths = allpop[j]
88          klas = paths[len(paths)-k:]
89          paths = paths[:len(paths)-k]
90          jh = np.empty((0, k), float)
91          f=0
92          ind =0
93
94          for s in range(len(klas)):
95              tourLen = 0
96              nextIn = ind+klas[s]
97              rute=paths[ind:nextIn]
98
99              tourLen += dmatrix[0, rute[0]]
100             for i in range(len(rute) - 1):
101                 tourLen += dmatrix[rute[i], rute[i + 1]]
102             tourLen += dmatrix[rute[-1], 0]
103
104             jh = np.append(jh, tourLen)
105             ind = nextIn
106         f = 100/np.sum(jh)
107
108         fitness = np.append(fitness,f)
109         jarak = np.append(jarak,[jh],axis=0)
110
111     return fitness, jarak
```

Kode Program 5.7 Hitung *Fitness*

Pembahasan untuk implementasi Kode Program 5.7 akan dijabarkan sebagai berikut:

1. Baris 82 melakukan deklarasi fungsi bernama *Hitung_Fitness* yang akan berisi proses untuk menghitung jarak dan *fitness* dari setiap populasi berdasarkan distance matrix yang sebelumnya sudah dibuat.
2. Baris 83-84 melakukan inisialisasi terhadap variabel *jarak* dan *fitness* untuk menyimpan hasil keseluruhan perhitungan jarak dan nilai *fitness.*
3. Baris 86 melakukan inisialisasi perulangan sebanyak populasi atau data yang akan dihitung jarak dan nilai *fitness*nya.
4. Baris 87-92 melakukan inisialisasi terhadap variabel *paths* yang digunakan untuk menyimpan rute yang akan dihitung jarak dan nilai *fitness*nya, *klas* untuk menyimpan pembagian rute untuk 5 hari kerja, *jh* untuk menyimpan jarak harian, *f* untuk menyimpan nilai *fitness* untuk 5 hari kerja, *ind* untuk menyimpan nilai indeks untuk pergantian hari dalam proses menghitung jarak harian.
5. Baris 94 melakukan inisialisasi perulangan sejumlah *klass,* yaitu 5 untuk 5 hari kerja.
6. Baris 95 melakukan inisialisasai variabel *tourLen* yang digunakan untuk menyimpan perhitungan jarak, *nextInd* untk proses pergantian hari dalam perhitungan, *rute* untuk rute harian yang akan dihitung jaraknya.

7. Baris 99-105 merupakan proses inti, yaitu proses perhitungan jarak harian. Hasil dari proses ini akan disimpan pada variabel *jh*. Kemudian untuk berganti ke hari berikutnya nilai *ind* diganti dengan *nextIn*
8. Baris 106 menghitung nilai *fitness* untuk 5 hari kerja, yaitu dengan cara 100 dibagi total seluruh jarak untuk 5 hari kerja. Hasilnya akan disimpan dalam variabel *f.*
9. Baris 108 dan 109 menyimpan hasil keseluruhan jarak dan nilai *fitness* ke dalam variabel *jarak* dan *fitness.*
10. Baris 111 mengembalikan nilai berupa variabel *fitness* dan *jarak* yang menyimpan hasil dari perhitungan *fitness* dan jarak seluruh data.

5.2.6 Implementasi Fungsi Algen

Pada bagian fungsi algen ini akan berisi proses utama dari Algoritma genetika, mulai dari inisialisasi jumlah *popsize*, nilai *cr* dan *mr*, *jmlGen*, inisialisasi kromosom, memanggil fungsi-fungsi yang sebelumnya sudah dijelaskan sebelumnya, hingga proses menghitung total jarak tempuh. Kode program untuk implementasi fungsi algen ini terdapat pada Kode Program 5.8.

Algoritma Genetika : Fungsi Algen

```
113 def algen(dmatrix):
114     popsize = 20
115     cr = 0.4
116     mr = 0.6
117     jmlGen = 8
118     k = 5
119 
120     w, h = np.shape(dmatrix)
121 
122     pop = np.empty((0, w+4), int)
123     arr = np.arange(w-1)+1
124 
125     part2=np.full(k,(len(dmatrix)-1)/k, int)
126     sisa = (len(dmatrix)-1)-np.sum(part2)
127 
128     for i in range(sisa):
129         part2[i]+=1
130 
131     arr = np.append(arr,part2)
132 
133     for i in range(popsize):
134         np.random.shuffle(arr[:len(dmatrix)-1])
135         pop = np.append(pop, [arr], axis=0)
136 
137     jml_childcross = round(popsize*cr)
138     jml_childmut = round(popsize*mr)
139 
140     for i in range(jmlGen):
141 
142         childcross = crossover(pop, jml_childcross,k)
143         childmutasi = mutasi(pop, jml_childmut,k)
144 
145         allpop = np.array(pop)
146         allpop = np.append(allpop, childcross, axis=0)
147         allpop = np.append(allpop, childmutasi, axis=0)
148         allpop = np.asarray(allpop)
149 
```

```
150         fitness,jarak = Hitung_fitness(allpop, dmatrix,k)
151
152         newpop = np.empty((0, 3))
153
154         for i in range(len(allpop)):
155             newpop = np.append(
156                 newpop, [[allpop[i],
                    np.sum(jarak[i]),100/np.sum(jarak[i])]], axis=0)
157
158         hasil = sorted(newpop, key=lambda x: x[1])
159
160         for i in range(popsize):
161             pop[i] = hasil[i][0]
162     ind=0
163     jrk=0
164     f=0
165     for i in range(len(part2)):
166         n = ind + part2[i]
167         print("  Hari ",i+1," : ",hasil[0][0][ind:n])
168         ind = n
169         jrk = hasil[0][1]
170         f = hasil[0][2]
171     print("-----------------------------------")
172     print("  Jarak = ", jrk, " km")
173     print("  Fitness = ", f)
174
175     return hasil[0]
```

Kode Program 5.8 Fungsi Algen

Pembahasan untuk implementasi Kode Program 5.8 akan dijabarkan sebagai berikut:

1. Baris 113 melakukan deklarasi fungsi bernama *algen*, di mana fungsi ini akan berisi main proses dari Algoritma genetika.
2. Baris 114-118 melakukan inisialisasi terhadap beberapa variabel yang menjadi parameter utama, yaitu *popsize* untuk jumlah populasi, *cr* untuk *crossover rate*, *mr* untuk *mutation rate*, *jmlGen* untuk jumlah generasi, dan *k* untuk jumlah hari kerja dalam seminggu (yang digunkana adalah 5 hari).
3. Baris 120 menyimpan ukuran baris dan kolom dari *distance matrix* untuk keperluan pembentukan kromosom.
4. Baris 122-135 membentuk kromosom sejumlah *popsize*, di mana kromosom dibentuk secara acak urutannya. Kemudian ditambahkan dengan kromosom bagian 2 yang merepresentasikan pembagian rute untuk 5 hari kerja. Pembagian dilakukan untuk setiap harinya hampir merata.
5. Baris 137 dan 138 melakukan inisialisasi jumlah *child* dari *crossover* dan mutasi. Jumlah *child* diperoleh dari proses perkalian *popsize* dengan *cr* untuk *crossover* dan perkalian popsize dengan *mr* untuk mutasi.
6. Baris 140 melakukan inisialisasi perulangan sebanyak jumlah gen pada variabel *jmlGen*.
7. Baris 142 memanggil fungsi *crossover* dengan parameter populasi sejumlah *popsize*, jumlah *child crossover,* dan *k*.

8. Baris 143 memanggil fungsi mutasi dengan parameter populasi sejumlah *popsize*, jumlah *child* mutasi, dan *k*.
9. Baris 145-148 merupakan proses penggabungan keseluruhan populasi, yaitu populasi awal, *child* hasil *crossover*, dan *child* hasil mutasi yang disimpan dalam variabel *allpop*.
10. Baris 150 memanggil fungsi *Hitung_fitness* dengan parameter *allpop*, *distance matrix,* dan *k*. Hasil dari proses ini akan disimpan di variabel *fitness* dan *jarak*.
11. Baris 152-158 merupakan proses seleksi, di mana *allpop* digabungkan dengan *jarak* dan *fitness*, kemudian akan diurutkan berdasarkan jarak mulai dari terkecil ke terbesar dan disimpan dalam variabel *hasil*.
12. Baris 160-161 melakukan menggainti nilai variabel *pop* sesuai dengan hasil seleksi. Proses ini berfungsi untuk memperbaharui populasi untuk proses generasi berikutnya.
13. Baris 162-173 merupakan proses untuk menampilkan *output* berupa rute harian untuk setiap *sales* beserta jarak tempuh dan nilai *fitness*nya dalam satu minggu.
14. Baris 175 mengembalikan nilai berupa variabel *hasil* pada indeks ke 0, di mana indeks ke 0 merupakan hasil terbaik dengan jarak terpendek dan *fitness* terbesar.

5.3 Implementasi Save to Excel

Pada bagian ini akan melakukan implementasi *save to excel*, di mana di dalamnya terdapat rangkaian proses untuk menyimpan hasil klasterisasi ke dalam file berformat excel. Kode program untuk implementasi generate *centroid* awal ini terdapat pada Kode Program 5.9.

```
    Save to Excel
1   def saveexcel(data,harian,nm_file):
2
3       writer = ExcelWriter("H:/KULIAH/Semester
              7/Skripsi/Skripsi/p2/coding + data/"+nm_file+".xlsx")
4       ind = 0
5
6       for i in range(len(harian)):
7           m = "Hari "+str(i+1)
8           nextIn = ind + harian[i]
9           hari = (pd.DataFrame({
10                      'No': data[ind:nextIn, 0],
11                      'Kode': data[ind:nextIn, 1],
12                      'Nama': data[ind:nextIn, 2],
13                      'Alamat': data[ind:nextIn, 3],
14                      'Latitude': data[ind:nextIn, 4],
15                      'Longitude': data[ind:nextIn, 5]}))
16          hari.to_excel(writer, m, index=False)
17          ind = nextIn
18      writer.save()
```

Kode Program 5.9 Implementasi *Save to Excel*

Pembahasan untuk implementasi Kode Program 5.9 akan dijabarkan sebagai berikut:

1. Baris 1 Mendefinisikan fungsi dengan nama *saveexcel* yang di dalamnya terdapat rangkaian proses untuk menyimpan hasil klasterisasi ke dalam file berformat excel.
2. Baris 3 menentukan *path* untk lokasi penyimpanan file *excel* yang disimpan dalam variabel *writer*.
3. Baris 4-18 merupakan proses untuk menyimpan data setiap klaster dari hasil proses optimasi rute ke dalam bentuk *dataframe* yang disimpan dalam variabel *hasil*.

5.4 Implementasi Main Program

Pada bagian ini akan melakukan implementasi main program, dimana seluruh fungsi untuk proses *K-means,* membuat matriks jarak, Algoritma Genetika, hingga menyimpan hasil ke dalam Excel akan digunakan. Kode program untuk implementasi generate *centroid* awal ini terdapat pada Kode Program 5.10.

Main Program

```
1   def main():
2       dataset = pd.read_excel(
3           "H:/KULIAH/Semester 7/Skripsi/Skripsi/p2/coding +
             data/datasetfull.xlsx", delimiter=";")
4       data = dataset.values
5       k=3
6       centroid, label = kmeans(data,k)
7
8       rute =[]
9
10      for i in range(k):
11          sales = np.empty((0,6))
12          cluster = data[np.where(label==i+1)]
13          cluster = np.insert(cluster, 0, [0, "Start Point", "PT
14          Indomarco Adi Prima","Jl lenjen suprapto no 70",
15          -7.616966,111.894101], axis=0)

16          print("=====================================")
17          print("Sales ",i+1," : ")
18          print("=====================================")
19          rute.append(algen(haversine(cluster[:, 4:])))
20          sales = np.append(sales, np.take(cluster,rute[i][0]
21                  [:len(rute[i][0])-5],axis=0),axis=0)
22          saveexcel(sales,rute[i][0][len(rute[i][0])-5:],"Rencana
                     Perjalanan Sales "+str(i+1))
        print("================================")
23      print("Total Jarak = ", rute[0][1]+rute[1][1]+rute[2][1]," km")
24      print("Fitness = ", 100 / (rute[0][1]+rute[1][1]+rute[2][1]))
25
26  if __name__ == "__main__":
27      main()
```

Kode Program 5.10 Implementasi *Main Program*

Pembahasan untuk implementasi Kode Program 5.9 akan dijabarkan sebagai berikut:

1. Baris 1 mendefinisikan fungsi dengan nama main.

2. Baris 2-4 melakukan inisialisasi dataset berupa data toko langganan dalam bentuk excel, dataset akan disimpan dalam variabel *data.*
3. Baris 5-6 melakukan inisialisasi nilai *k* yang merupakan banyak klaster yang akan dibentuk, kemudian memanggil fungsi *kmeans* dengan parameter *data* dan *k* untuk melakukan proses pengelompokan data. Kemudian hasil pengelompokan disimpan dalam variabel *centroid* dan label.
4. Baris 8-24 memanggil fungsi haversine dan *algen* dengan parameter data hasil pengelompokan untuk membuat matriks jarak sekaligus mencari rute paling optimal. Hasil rute optimal untuk seluruh klaster akan disimpan dalam variabel *rute.* Pada proses ini akan sekaligus menyimpan hasil optimasi rute 5 hari kerja untuk setiap *sales* ke dalam file *excel*. Selain itu juga akan menampilkan *output* berupa urutan rute harian untuk setiap klaster dari hasil optimasi yang disertai dengan total jarak tempuh beserta nilai *fitness*nya.
5. Baris 26-27 syntax muntuk menjalankan fungsi main.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bagian ini akan dilakukan pengujian sesuai dengan perancangan pengujian yang telah dibuat pada bab sebelumnya. Pengujian ini dilakukan untuk mengetahui kualitas klaster yang terbentuk dan nilai parameter yang paling optimal dalam kasus distribusi produk PT Indomarco Adi Prima *Stock Point* Nganjuk. Selain itu juga dilakukan pengujian perbandingan hasil dari gabungan dari Algoritma K-Means dan Algoritma Genetika dengan Algoritma Genetika tanpa K-Means serta dengan rute asli yang digunakan perusahaan. Parameter yang diuji adalah ukuran populasi, banyak generasi, *crossover rate,* dan *mutation rate*. Hasil akhir yang diperoleh dari pengujian ini adalah berupa rute yang paling optimal untuk 5 hari kerja dalam satu minggu dengan asumsi bahwa setiap hari nya titik keberangkatan dan akhir perjalanan *salesman* adalah di PT Indomarco Adi Prima (Stock Point Nganjuk).

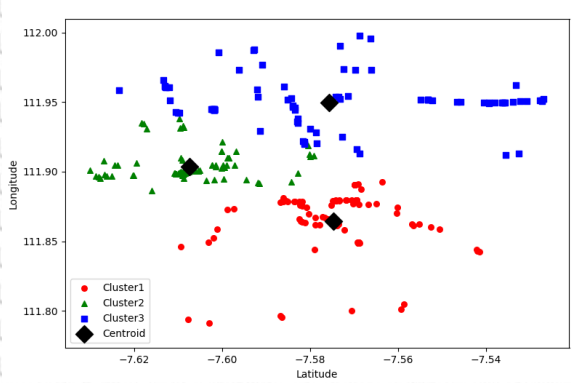
6.1 Pengujian Kualitas Klaster

Pengujian kualitas klaster dalam penelitian ini bertujuan untuk mengetahui kualitas dari klaster yang peroleh dari hasil *K-Means* dan membandingkannya dengan data pembagian oleh yang digunakan perusahaan saat ini. Pada pengujian ini, yang akan dibandingkan adalah nilai *Silhouette Coefficient* yang dihasilkan oleh klaster yang diperoleh hasil dari *K-Means* dengan nilai *Silhouette Coefficient* dari data yang digunakan perusahaan saat ini. Pengujian dilakukan sebanyak 3 kali dengan jumlah data yang sama yaitu 275 dengan banyak klaster adalah 3. Hasil pengujian dapat dilihat pada Tabel 6.1:

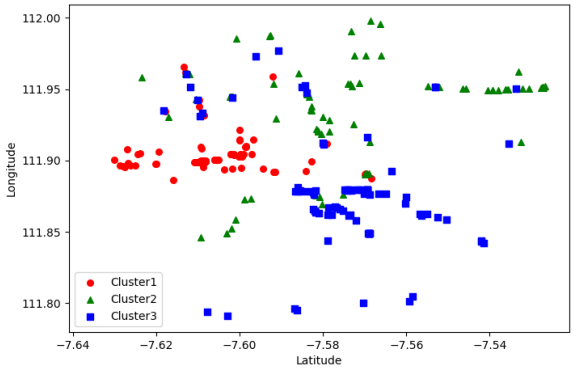
Tabel 6.1 Hasil Pengujian Kualitas Klaster

NO	Jumlah Data	Banyak Klaster	Nilai *Silhouette Coefficient*	
			Hasil *K-Means*	Data asli tanpa K-Means
1	275	3	0,471743	0,267071
2	275	3	0,471743	0,267071
3	275	3	0,471743	0,267071

Dari Tabel 6.1 dapat di lihat bahwa nilai *Silhouette Coefficient* dari hasil *K-Means* adalah stabil di 0,471743 sedangkan nilai *Silhouette Coefficient* dari data dari perusahaan adalah 0,267071. hasil pembagian dari *K-Means* lebih baik dari data asli perusahaan yang digunakan saat ini, karena nilai *Silhouette Coefficient* hasil *K-Means* lebih baik karena lebih besar dan mendekati 1. Hal tersebut dapat dibuktikan dari Gambar 6.1 dan Gambar 6.2.



Gambar 6.1 Visualisasi Hasil Klaster *K-Means*



Gambar 6.2 Visualisasi Data Dari Perusahaan

Dari Gambar 6.1 dan Gambar 6.2 dapat dilihat bahwa keduanya memiliki perbedaan, di mana pada Gambar 6.1 setiap klaster hampir terpisah dengan baik dari klaster lainnya, sedangkan pada Gambar 6.2 banyak sekali titik-titik yang melenceng jauh dari klasternya. Dapat disimpulkan bahwa metode K-Means dapat diterapkan dalam permasalahan pada penelitian ini. Hasil dari K-Means pada penilitian kali ini adalah 3 klaster yang merepresentasikan titik yang harus dikunjungi dalam satu minggu oleh salesman yang berjumlah tiga orang. Namun pada perusahaan PT Indomarco Adi Prima (Stock Point Nganjuk) hanya berlaku 5 hari efektif kerja, sehingga perlu dilakukan pembagian lagi dari data mingguan setiap sales menjadi data titik kunjungan harian. Proses tersebut akan dilakukan dengan menggunakan Algoritma Genetika untuk menghasilkan rute paling optimal selama 5 hari kerja dalam satu minggu.

6.2 Pengujian Ukuran Populasi

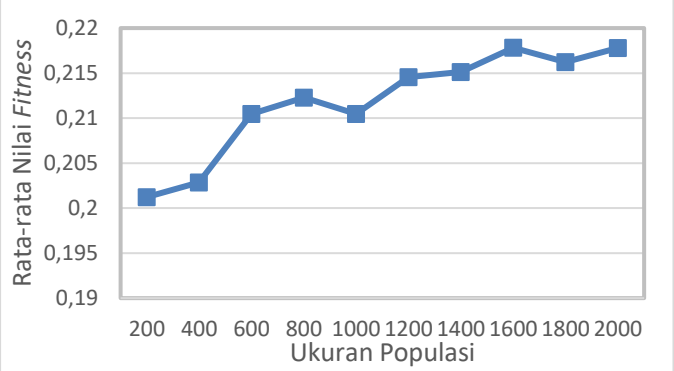
Pengujian ukuran populasi dalam penelitian ini adalah bertujuan untuk mengukur dan melihat pengaruh ukuran populasi terhadap nilai *fitness* yang dihasilkan. Pengujian ukuran populasi yang dilakukan adalah dengan menggunakan ukuran populasi kelipatan 200 sampai 2000 dengan nilai *crossover rate* (*cr*=0,6), *mutation rate* (*mr*=0,4), dan jumlah generasi (*jmlGen*=100). Setiap ukuran populasi akan dilakukan pengujian sebanyak 10 kali. Berikut hasil dari

pengujian ukuran populasi terdapat pada bagian Tabel 6.2 dan hasil tabel pengujian secara detail terdapat pada Lampiran B.

Tabel 6.2 Hasil Pengujian Ukuran Populasi

Ukuran populasi	*Fitness* Percobaan ke-						Rata-rata *Fitness*
	1	2	3	…	9	10	
200	0,204786	0,202917	0,203000	…	0,197242	0,199793	0,201229
400	0,211157	0,202551	0,196434	…	0,193142	0,207205	0,202852
600	0,217374	0,203787	0,199798	…	0,210123	0,210428	0,21046
800	0,217167	0,210345	0,207360	…	0,218015	0,205555	0,212271
1000	0,211382	0,217325	0,209558	…	0,209446	0,209056	0,210479
1200	0,220762	0,221302	0,208154	…	0,213213	0,214266	0,214573
1400	0,219044	0,215750	0,213545	…	0,217293	0,215596	0,215108
1600	0,212504	0,210993	0,219184	…	0,211407	0,210484	0,217810
1800	0,213919	0,236758	0,222363	…	0,215528	0,220891	0,216245
2000	0,20447	0,202433	0,227072	…	0,217955	0,232638	0,217797

Dari Tabel 6.2 didapatkan nilai rata-rata *fitness* yang dihasilkan dari percobaan sebanyak 10 kali pada penguijian ukuran populasi dan akan ditampilkan dalam bentuk diagram garis. Diagram garis untuk rata-rata nilai *fitness* dari hasil pengujian ukuran populasi dapat dilihat pada Gambar 6.3.



Gambar 6.3 Hasil Pengujian Ukuran Populasi

Dari hasil pengujian seperti pada Tabel 6.2 dan grafik hasil pengujian pada Gambar 6.3 dapat dilihat bahwa hasil rata-rata nilai *fitness* terkecil yaitu pada saat populasi berukuran 200 dengan nilai *fitness* yang dihasilkan sebesar 0,201229. Sedangkan rata-rata nilai *fitness* terbesar adalah Ketika populasi berukuran 1600 dengan nilai *fitness* yang dihasilkan adalah sebesar 0,217810. Dari pergerakan grafik terlihat terjadi peningkatan secara terus menerus pada populasi 200 sampai 800 dan pada ukuran populasi 1000 mengalami penurunan, kemudan pada populasi 1200 menunjukkan kenaikan Kembali. Dan pada populasi 1800 mengalami sedikit penurunan, dan setelahnya menunjukkan keadaan rata-rata *fitness* yang stabil. Dari grafik pengujian di atas dapat disimpulkan bahwa ukuran populasi berpengaruh terhadap nilai *fitness*. Semakin banyak ukuran populasi akan membuat variasi solusi yang dihasilkan semakin beragam, sehingga tidak

menutup kemungkinan akan dapat menghasilkan solusi dengan nilai *fitness* yang tinggi. Pada kasus dengan titik kunjungan sebanyak 275, dengan jumlah *sales* adalah 3, percobaaan dilakukan sebanyak 10 kali dengan nilai *cr*=0,6 dan *mr*=0,4 serta jumlah generasi (*jmlGen*=100), algoritma gabungan K-Means dan Algoritma Genetika menghasilkan solusi terbaik pada jumlah populasi 1600.

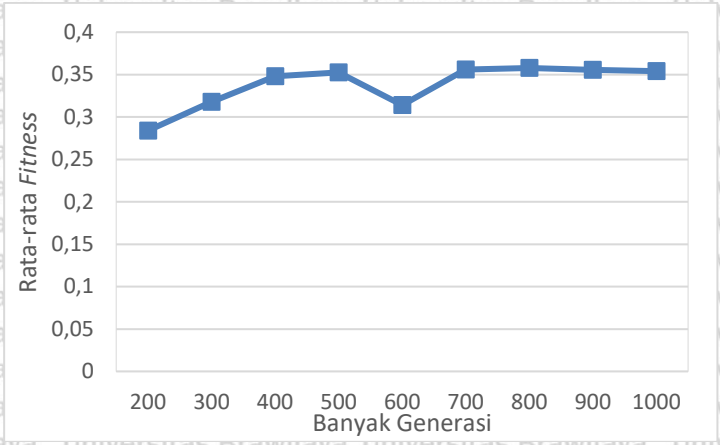
6.3 Pengujian Banyak Generasi

Pada pengujian banyak generasi ini akan dilakukan pengujian untuk mengetahui jumlah generasi yang optimal dalam pencarian solusi. Jumlah generasi yang akan digunakan dalam pengujian ini adalah kelipatan 100 yang dimulai dari 200 hingga 1000. Pengujian banyak generasi ini dilakukan berdasarkan ukuran populasi terbaik dari hasil pengujian sebelumnya, yaitu 1600 dengan nilai *crossover rate* (*cr*=0,6) dan *mutation rate* (*mr*=0,4), pengujian pada setiap generasi akan dilakukan sebanyak 10 kali. Berikut nilai dari pengujian banyak generasi terdapat pada bagian Tabel 6.3 dan hasil tabel pengujian secara detail terdapat pada Lampiran C.

Tabel 6.3 Hasil Pengujian Banyak Generasi

Banyak Generasi	*Fitness* Percobaan ke-						Rata-rata *Fitness*
	1	2	3	...	9	10	
200	0,297581	0,278075	0,275523	...	0,285447	0,275996	0,283988
300	0,310883	0,335352	0,323795	...	0,315223	0,312422	0,317685
400	0,341665	0,344867	0,350723	...	0,364026	0,348084	0,347881
500	0,363987	0,349730	0,342058	...	0,366722	0,341862	0,352562
600	0,311415	0,311894	0,315305	...	0,317028	0,317967	0,314003
700	0,351719	0,350082	0,338127	...	0,354126	0,358713	0,356142
800	0,354499	0,349074	0,363156	...	0,368319	0,345329	0,357828
900	0,352246	0,348129	0,369242	...	0,342948	0,345606	0,355562
1000	0,339057	0,342257	0,336115	...	0,356803	0,365150	0,354083

Dari Tabel 6.3 didapatkan nilai rata-rata *fitness* yang dihasilkan dari percobaan sebanyak 10 kali pada penguijian banyak generasi dan akan ditampilkan dalam bentuk diagram garis. Diagram garis untuk rata-rata nilai *fitness* dari hasil pengujian banyak generasi dapat dilihat pada Gambar 6.4.



Gambar 6.4 Hasil Pengujian Banyak Generasi

Dari grafik hasil pengujian banyak generasi pada Gambar 6.4, menunjukkan bahwa rata-rata nilai *fitness* terendah berada pada jumlah generasi 200 dengan rata-rata nilai *fitness* 0,283988. Sedangkan rata-rata nilai *fitness* tertinggi diperoleh pada saat jumlah generasi 800 dengan nilai rata-rata *fitness* 0,357828. Grafik hasil pengujian banyak generasi menunjukkan bahwa grafik terus mengalami kenaikan, yaitu terjadi pada saat pengujian sebanyak 200 sampai 500 generasi dan pada generasi 600 sedikit mengalami penurunan, dan pada generasi 700 ke atas menunjukkan keadaan rata-rata nilai *fitness* yang cukup stabil. Terjadinya kenaikan dan penurunan rata-rata nilai *fitness* pada setiap pengujian banyak generasi disebabkan karena kromosom diperoleh secara acak, sehingga memungkinkan jika nilai *fitness* yang diperoleh pada kromosom lebih kecil ataupun lebih besar. Pada kasus dengan titik kunjungan sebanyak 275, dengan jumlah *sales* adalah 3, percobaaan dilakukan sebanyak 10 kali dengan nilai nilai *crossover rate* (*cr*=0,6), *mutation rate* (*mr*=0,4), dan ukuran populasi (*popSize*=1600), *K-Means* dan Algoritma Genetika menghasilkan solusi terbaik pada banyak generasi 800.

6.4 Pengujian Kombinasi *Crossover rate dan Mutation Rate*

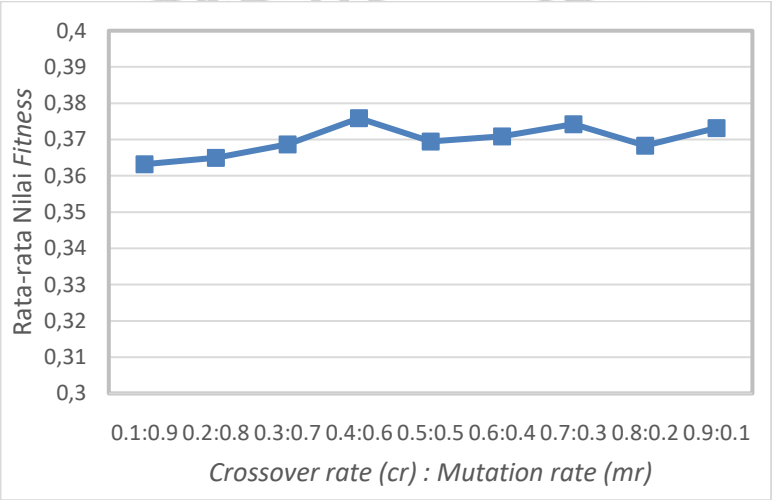
Pengujian kombinasi *crossover rate (cr)* dan *mutation rate (mr)* dilakukan untuk mengetahui kombinasi *cr* dan *mr* yang optimal dalam pencarian solusi serta melihat pengaruhnya terhadap nilai *fitness* yang dihasilkan. Kombinasi *cr* dan *mr* yang digunakan dalam pengujian kali ini yaitu 0,1:0,9; 0,2:0,8; 0,3:0,7; 0,4:0,6; 0,5:0,5; 0,6:0,4; 0,7:0,3; 0,8:0,2; 0,9:0,1. Pengujian kombinasi *cr* dan *mr* ini dilakukan berdasarkan ukuran populasi terbaik dari hasil pengujian sebelumnya, yaitu 1600 dengan jumlah generasi adalah 800, di mana pengujian pada setiap kombinasi akan dilakukan sebanyak 10 kali. Berikut nilai dari pengujian kombinasi *cr* dan *mr* terdapat pada bagian Tabel 6.4 dan hasil pengujian secara detail terdapat pada Lampiran D.

Tabel 6.4 Hasil Pengujian Kombinasi *Crossover rate dan Mutation Rate*

cr : *mr*	*Fitness* Percobaan ke-						Rata-rata *Fitness*
	1	2	3	...	9	10	

0,1:0,9	0,373691	0,367506	0,364649	...	0,364338	0,332655	0,363201
0,2:0,8	0,360465	0,359117	0,358174	...	0,365761	0,374081	0,365002
0,3:0,7	0,364291	0,367496	0,365584	...	0,361806	0,355515	0,368657
0,4:0,6	0,384198	0,366677	0,368464	...	0,383839	0,369551	0,375874
0,5:0,5	0,373712	0,379843	0,373815	...	0,367249	0,364821	0,369498
0,6:0,4	0,373691	0,380074	0,363962	...	0,373187	0,360628	0,370853
0,7:0,3	0,378840	0,378382	0,366438	...	0,377212	0,379898	0,374239
0,8:0,2	0,378695	0,375679	0,376522	...	0,366315	0,362314	0,368317
0,9:0,1	0,356016	0,381787	0,387604	...	0,375230	0,367744	0,373167

Dari Tabel 6.4 didapatkan nilai rata-rata *fitness* yang dihasilkan dari percobaan sebanyak 10 kali pada penguijian kombinasi *cr* dan *mr* yang akan ditampilkan dalam bentuk diagram garis. Diagram garis untuk rata-rata nilai *fitness* dari hasil pengujian kombinasi *cr* dan *mr* dapat dilihat pada Gambar 6.5.



Gambar 6.5 Hasil Pengujian Kombinasi *Crossover rate dan Mutation Rate*

Dari grafik hasil pengujian kombinasi *cr* dan *mr* pada Gambar 6.5, menunjukkan bahwa nilai *fitness* dengan rata-rata terendah adalah ketika kombinasi *cr* 0,1 dan *mr* 0,9 dengan rata-rata nilai *fitness* 0,363201. Sedangkan rata-rata *fitness* tertinggi adalah ketika kombinasi *cr* 0,4 dan *mr* 0,6 dengan rata-rata nilai *fitness* 0,375874. Pada grafik hasil pengujian kombinasi *cr* dan *mr* dapat dilihat bahwa rata-rata nilai *fitness* cenderung lebih baik ketika nilai *cr* dan *mr* mendekati nilai seimbang. Sebab, apabila nilai *cr* lebih kecil dari *mr* akan menyebabkan proses *crossover* tidak bisa maksimal dalam memanfaatkan informasi yang ada pada generasi sebelumnya untuk menghasilkan individu yang lebih baik. Dan sebaliknya, apabila nilai *mr* lebih kecil dari *cr* akan menyebabkan proses modifikasi individu untuk menambah keragaman populasi akan kurang maksimal, meskipun proses *crossover* dapat maksimal, namun individu yang dihasilkan akan kurang bervariasi. Pada kasus dengan titik kunjungan sebanyak 275, dengan jumlah *sales* adalah 3, percobaaan dilakukan sebanyak 10 kali dengan

ukuran populasi 1600 dan banyak generasi adalah 800, algoritma *K-Means* dan Algoritma Genetika dapat menghasilkan solusi terbaik pada kombinasi *crossover rate* 0,4 dan *mutation rate* 0,6.

6.5 Pengujian Perbandingan Gabungan Metode *K-Means* dan Algoritma Genetika (GKA) dengan Algoritma Genetika tanpa *K-Means*

Pengujian perbandingan gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan Algoritma Genetika tanpa *K-Means* ini bertujuan untuk membandingkan solusi yang diperoleh antara keduanya. Pengujian perbandingan antara kedua algoritma ini dilakukan berdasarkan parameter yang sama, yaitu ukuran populasi terbaik yang menghasilkan *fitness* terbesar dari hasil pengujian sebelumnya, yaitu 1600 dengan jumlah generasi adalah 800 serta kombinasi *crossover rate* 0,4 dan *mutation rate* 0,6. Untuk solusi yang dihasilkan oleh GKA akan diambilkan dari hasil terbaik yang diperoleh dari keseluruhan pengujian sebelumnya. Hasil optimalisasi rute terpilih dengan menggunakan metode gabungan *K-Means* dan Algoritma Genetika berdasarkan pengujian sebelumnya adalah sebagai berikut:

Rute *Sales* 1:

Hari 1 = [20, 90, 14, 91, 56, 54, 53, 23, 82, 22, 58, 86, 46, 21, 10, 60, 9, 13, 88]

Hari 2 = [69, 68, 12, 89, 25, 73, 76, 67, 71, 64, 63, 70, 79, 3, 6, 2, 4, 1, 7]

Hari 3 = [5, 92, 81, 62, 74, 80, 61, 83, 72, 85, 66, 65, 75, 78, 84, 77, 18, 8]

Hari 4 = [43, 40, 45, 51, 52, 41, 42, 50, 49, 47, 57, 35, 55, 39, 37, 48, 44, 33]

Hari 5 = [26, 24, 38, 36, 34, 59, 29, 32, 27, 87, 30, 31, 28, 19, 15, 17, 16, 11]

Dengan total jarak yang dihasilkan adalah 95,960338 km.

Rute *Sales* 2:

Hari 1 = [63, 48, 10, 21, 71, 44, 69, 87, 86, 72, 85, 90, 84, 91, 89, 67, 74, 94, 80]

Hari 2 = [92, 81, 93, 82, 88, 59, 75, 79, 77, 78, 52, 20, 76, 68, 73, 50, 66, 37, 49]

Hari 3 = [35, 42, 43, 41, 16, 1, 2, 45, 46, 22, 27, 28, 39, 25, 26, 36, 29, 38, 70]

Hari 4 = [18, 19, 15, 4, 13, 11, 6, 23, 34, 5, 14, 8, 12, 9, 32, 31, 7, 33, 17]

Hari 5 = [24, 47, 30, 54, 62, 56, 57, 58, 55, 40, 60, 53, 61, 65, 64, 83, 51, 3]

Dengan total jarak yang dihasilkan adalah 44,689992 km.

Rute *Sales* 3:

Hari 1 = [71, 72, 25, 69, 68, 73, 17, 66, 65, 58, 36, 47, 80, 79, 89, 7, 2, 49]

Hari 2 = [62, 51, 59, 60, 55, 54, 56, 88, 64, 57, 53, 52, 63, 48, 1, 76, 39, 85]

Hari 3 = [6, 81, 41, 42, 40, 3, 4, 86, 83, 16, 45, 5, 74, 38, 23, 26, 44, 43]

Hari 4 = [32, 20, 34, 24, 31, 22, 82, 77, 11, 33, 14, 30, 10, 50, 12, 13, 21, 29]

Hari 5 = [28, 87, 27, 35, 61, 67, 18, 8, 37, 19, 46, 78, 75, 84, 15, 9, 70]

Dengan total jarak yang dihasilkan adalah 119,072007 km.

Sehingga diperoleh total jarak dari ketiga *sales* adalah 259,722337 km dengan *fitness* 0,385026. Dari hasil rute di atas diasumsikan bahwa setiap hari *salesman* mengawali dan mengakhiri perjalanan di PT Indomarco Adi Prima (Stock Point Nganjuk). Jadi setelah menyelesaikan kunjungan pada hari pertama, untuk hari kedua akan kembali memulai dan mengakhiri perjalanan di PT Indomarco Adi Prima (Stock Point Nganjuk), begitu juga untuk hari-hari berikutnya. Untuk Hasil dari program dapat di lihat pada Gambar 6.6.

```
================================================
Sales 1 :
================================================
  Hari  1  :  [20, 90, 14, 91, 56, 54, 53, 23, 82, 22, 58, 86, 46, 21, 10, 60, 9, 13, 88]
  Hari  2  :  [69, 68, 12, 89, 25, 73, 76, 67, 71, 64, 63, 70, 79, 3, 6, 2, 4, 1, 7]
  Hari  3  :  [5, 92, 81, 62, 74, 80, 61, 83, 72, 85, 66, 65, 75, 78, 84, 77, 18, 8]
  Hari  4  :  [43, 40, 45, 51, 52, 41, 42, 50, 49, 47, 57, 35, 55, 39, 37, 48, 44, 33]
  Hari  5  :  [26, 24, 38, 36, 34, 59, 29, 32, 27, 87, 30, 31, 28, 19, 15, 17, 16, 11]
------------------------------------
  Jarak =  95.96033815140026 km
  Fitness =  1.0420972031405955
================================================
Sales 2 :
================================================
  Hari  1  :  [63, 48, 10, 21, 71, 44, 69, 87, 86, 72, 85, 90, 84, 91, 89, 67, 74, 94, 80]
  Hari  2  :  [92, 81, 93, 82, 88, 59, 75, 79, 77, 78, 52, 20, 76, 68, 73, 50, 66, 37, 49]
  Hari  3  :  [35, 42, 43, 41, 16, 1, 2, 45, 46, 22, 27, 28, 39, 25, 26, 36, 29, 38, 70]
  Hari  4  :  [18, 19, 15, 4, 13, 11, 6, 23, 34, 5, 14, 8, 12, 9, 32, 31, 7, 33, 17]
  Hari  5  :  [24, 47, 30, 54, 62, 56, 57, 58, 55, 40, 60, 53, 61, 65, 64, 83, 51, 3]
------------------------------------
  Jarak =  44.6899920703885 km
  Fitness =  2.237637452306907
================================================
Sales 3 :
================================================
  Hari  1  :  [71, 72, 25, 69, 68, 73, 17, 66, 65, 58, 36, 47, 80, 79, 89, 7, 2, 49]
  Hari  2  :  [62, 51, 59, 60, 55, 54, 56, 88, 64, 57, 53, 52, 63, 48, 1, 76, 39, 85]
  Hari  3  :  [6, 81, 41, 42, 40, 3, 4, 86, 83, 16, 45, 5, 74, 38, 23, 26, 44, 43]
  Hari  4  :  [32, 20, 34, 24, 31, 22, 82, 77, 11, 33, 14, 30, 10, 50, 12, 13, 21, 29]
  Hari  5  :  [28, 87, 27, 35, 61, 67, 18, 8, 37, 19, 46, 78, 75, 84, 15, 9, 70]
------------------------------------
  Jarak =  119.07200728763448 km
  Fitness =  0.8398279518244496
================================================
Total Jarak =  259.72233750942326 km
Fitness =  0.3850265670598001 5
```

Gambar 6.6 Hasil *Running* Program GKA

Kemudian pengujian menggunakan metode GA tanpa digabung dengan *K-Means* dilakukan dengan menggunakan parameter yang sama dengan metode GKA, yaitu diambil dari hasil terbaik dengan ukuran populasi 1600 dengan jumlah generasi adalah 800 serta kombinasi *crossover rate* 0,4 dan *mutation rate* 0,6. Untuk jumlah titik kunjungan pada setiap *sales* juga sama dengan pembagian K-Means, yaitu masing-masing *sales* 1, *sales* 2 dan *sales* 3 adalah 92, 94, dan 89. Untuk hasil optimasi rute dengan menggunakan Algoritma Genetika tanpa digabung dengan *K-Means* adalah sebagai berikut:

Rute *Sales* 1:

Hari 1 = [184, 50, 73, 20, 67, 103, 166, 161, 97, 110, 137, 126, 111, 172, 211, 70, 17, 33, 49]

Hari 2 = [79, 164, 165, 86, 52, 68, 4, 45, 51, 57, 61, 64, 53, 65, 85, 60, 9, 25, 32]

Hari 3 = [18, 12, 42, 14, 144, 272, 81, 59, 76, 1, 24, 48, 3, 2, 41, 29, 27, 22]

Hari 4 = [71, 78, 43, 19, 47, 8, 7, 271, 75, 37, 77, 66, 28, 38, 13, 54, 10, 92]

Hari 5 = [74, 150, 134, 91, 218, 82, 159, 167, 56, 11, 40, 58, 6, 34, 31, 39, 36, 46]

Rute *Sales* 2:

Hari 1 = [221, 140, 214, 105, 108, 124, 121, 101, 224, 131, 219, 112, 162, 122, 156, 113, 129, 115]

Hari 2 = [107, 106, 127, 125, 118, 120, 114, 216, 83, 153, 147, 142, 152, 157, 148, 273, 212, 149]
Hari 3 = [163, 141, 138, 220, 223, 171, 132, 117, 145, 123, 104, 116, 213, 128, 275, 143, 133, 222]
Hari 4 = [154, 151, 139, 95, 94, 136, 135, 155, 274, 80, 96, 84, 158, 215, 146, 160, 109, 168]
Hari 5 = [69, 16, 44, 93, 238, 21, 35, 182, 229, 183, 180, 268, 246, 89, 87, 174, 254, 225]

Rute *Sales* 3:
Hari 1 = [256, 185, 234, 173, 240, 267, 217, 169, 119, 130, 63, 55, 62, 15, 30, 23, 5, 26, 175]
Hari 2 = [259, 189, 88, 102, 100, 177, 209, 205, 199, 202, 258, 252, 196, 191, 193, 250, 261, 198, 201]
Hari 3 = [179, 181, 208, 263, 270, 99, 72, 236, 248, 264, 228, 227, 243, 200, 226, 244, 233, 210, 176]
Hari 4 = [269, 187, 231, 170, 266, 178, 247, 235, 230, 204, 206, 190, 237, 265, 192, 195, 194, 239]
Hari 5 = [241, 253, 245, 98, 251, 257, 249, 90, 232, 260, 186, 207, 255, 242, 262, 203, 188, 197]

Dari hasil rute di atas diperoleh total jarak dari ketiga *sales* adalah 1344,354477 km dengan *fitness* 0,074385. Hasil dari program dapat di lihat pada Gambar 6.7.

```
=====================================================
Sales  1  :
=====================================================
  Hari  1  :  [11 21 69 78 81 65 74 80 85 62 66 50  8 43 45 51 52 57 54]
  Hari  2  :  [90 91  9 60 13 77 76 73 25 64 71  5  2  7  6 92 75  1  4]
  Hari  3  :  [19 17 15 59 28 31 30 87 27 32 29 72 61 83 37 47 39 49]
  Hari  4  :  [23 14 53 46 82 12  3 68 88 18 84 79 67 63 70 89 86 20]
  Hari  5  :  [56 22 58 10 42 55 48 34 24 36 44 26 33 38 35 41 40 16]
-----------------------------------
  Jarak =  112.0280158549604  km
  Fitness =  0.8926338580294706
=====================================================
Sales  2  :
=====================================================
  Hari  1  :  [22 19 28 36 25 26  5 57 58 55 40 60 61 65 64 51 83 53  3]
  Hari  2  :  [30 34 45 41 42 16 69 86 87 72 85 84 91 90 89 67 75 56 62]
  Hari  3  :  [71 21 44 52 68 76 50 37 49 73 43 17 38 29 18 70 27 33 12]
  Hari  4  :  [ 2 11 46 39 24 23 32 13 15  8 31 14 10 47  6  4 54  7 48]
  Hari  5  :  [63 88 82 93 81 92 80 94 74 59 78 77 79 20 66 35  1  9]
-----------------------------------
  Jarak =  46.64926516584409  km
  Fitness =  2.1436564894320895
=====================================================
Sales  3  :
=====================================================
  Hari  1  :  [71 66 58 24 11 82 12 27  8 29 33 50 30 22 31 79 47 36]
  Hari  2  :  [73  7 43 74 23 38 83 86 16  5 45 44 26 20 34 32  2 80]
  Hari  3  :  [70 72 84  9 25 15 69 68 65 55 88 51 52 48 64 62 76 56]
  Hari  4  :  [57 63 54 53 49 89 60 59  6 41 81  3 40  4 42 85  1 39]
  Hari  5  :  [17 77 14 10 35 61 19 18 67 37 21 87 13 28 46 78 75]
-----------------------------------
  Jarak =  141.93402451754815  km
  Fitness =  0.7045526986211569
=====================================================
Total Jarak =  300.61130553835267  km
Fitness =  0.3326554861964158
```

Gambar 6.7 Hasil *Running* Program GA

Perbandingan hasil antara metode gabungan *K-Means* dan Algoritma Genetika dengan Algoritma Genetika tanpa digabung dengan *K-means* dapat di lihat pada Tabel 6.5.

Tabel 6.5 Perbandingan hasil metode GKA dan GA

Algoritma	Gabungan K-Means dan GA	GA Tanpa K-Means
Jumlah titik kunjungan	275	275
Jumlah *Sales*	3	3
Jumlah populasi	1600	1600
Jumlah generasi	800	800
Total Jarak (km)	259,722337	1344,354477
Fitness	0,385026	0,074385

Dari Tabel 6.5 dapat dilihat bahwa terdapat selisih antara hasil GKA dengan GA. Dari metode GKA menghasilkan total jarak 259,722337 km dengan *fitness* 0,385026. Sedangakan metode GA menghasilkan total jarak 1344,354477 km dengan *fitness* 0,074385. Sehingga dapat disimpulkan bahwa metode gabungan *K-Means* dan Algoritma Genetika lebih baik dari pada Algoritma Genetika tanpa digabung dengan *K-Means* pada studi kasus distribusi produk PT Indomarco Adi Prima (*Stock Point* Nganjuk) dengan selisih jarak 1084,63214 km.

6.6 Pengujian Perbandingan Gabungan Getode *K-Means* dan Algoritma Genetika (GKA) dengan Rute yang Digunakan Perusahaan

Pada pengujian ini akan dilakukan perbandingan dari hasil yang diperoleh dari gabungan metode *K-Means* dan Algoritma Genetika (GKA) dengan rencana perjalanan *sales* yang digunakan oleh PT Indomarco Adi Prima (*Stock Point Nganjuk*) saat ini. GKA akan menggunakan hasil terbaik dari pengujian yang sebelumnya dilakukan, sedangkan rute asli akan menghitung total jarak tempuh dari rencana perjalanan *sales* yang ada. kemudian akan dihitung selisih jarak tempuh antara hasil yang diperoleh dari GKA dengan rute asli yang digunakan saat ini. Untuk metode GKA menghasilkan total jarak 259,722337 km dengan *fitness* 0,385026. Sedangkan total jarak dari rencana perjalanan *sales* dapat dilihat pada Gambar 6.8.

```
==================================================
Sales 1 :
==================================================
  Hari 1 = [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
  Hari 2 = [20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38]
  Hari 3 = [39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56]
  Hari 4 = [57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74]
  Hari 5 = [75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 19]
----------------------------------
  Jarak =  371.0916953674871 km
  Fitness =  0.2694751762120986
==================================================
Sales 2 :
==================================================
  Hari 1 = [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18]
  Hari 2 = [19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36]
  Hari 3 = [37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54]
  Hari 4 = [55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72]
  Hari 5 = [73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90]
----------------------------------
  Jarak =  352.49940990079193 km
  Fitness =  0.28368841816825785
==================================================
Sales 3 :
==================================================
  Hari 1 = [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
  Hari 2 = [20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38]
  Hari 3 = [39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57]
  Hari 4 = [58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75]
  Hari 5 = [76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 19 19]
----------------------------------
  Jarak =  389.928990139574 km
  Fitness =  0.2564569512110533
==================================================
Total Jarak =  1113.5200954078532 km
Fitness =  0.08980529441040094
```

Gambar 6.8 Hasil Perhitungan Rute Asli

Dari Gambar 6.8 dapat dilihat bahwa terdapat selisih antara total jarak tempuh dari rute yang dihasilkan GKA dengan total jarak tempuh dari rencana perjalanan *sales* yang digunakan oleh perusahaan saat ini. Dari metode GKA menghasilkan total jarak 259,722337 km. Sedangakan rute rencana perjalanan *sales* menghasilkan total jarak 1113,520095 km. Sehingga dapat disimpulkan bahwa gabungan K-Means dan Algoritma Genetika dapat menghasilkan jarak tempuh yang lebih baik dengan selisih jarak 854,520095 km.

BAB 7 PENUTUP

Pada bab penutup ini akan dijelaskan mengenai kesimpulan dan saran dari penelitian dengan studi kasus optimasi rute distribusi produk PT Indomarco Adi Prima (Stock Point Nganjuk) dengan menggunakan Metode K-Means dan Algoritma Genetika (GKA).

7.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan yang disertai dengan hasil dan analisis pengujian pada bab sebelumnya, maka kesimpulan yang diperoleh dari penelitian pada studi kasus distribusi produk PT Indomarco Adi Prima (*Stock Point* Nganjuk) adalah sebagai berikut:

1. Gabungan metode *K-Means* dan Algoritma Genetika (GKA) ini dapat diimplementasikan untuk studi kasus optimasi rute distribusi produk PT Indomarco Adi Prima (*Stock Point* Nganjuk) yang dapat menghasilkan nilai parameter yang optimal melalui pengujian beberapa parameter.
2. Pada pengujian kualitas klaster, jumlah klaster yang digunakan dalam penelitian ini disesuaikan dengan jumlah kendaraan (*sales*) yang dimiliki oleh perusahaan yaitu 3. Nilai *Silhouette Coefficient* yang dihasilkan dari *K-Means* adalah 0,471743, sedangkan nilai *Silhouette Coefficient* dari data dari perusahaan adalah 0,267071. Hasil dari *K-means* lebih baik karena nilai *Silhouette Coefficient* lebih besar dan mendekati 1.
3. Pada pengujian ukuran populasi menghasilkan solusi terbaik pada ukuran populasi sebanyak 1600 dengan rata-rata nilai *fitness* 0,217810. Pada pengujian banyak generasi menghasilkan solusi terbaik ketika jumlah generasi 800 dengan rata-rata nilai *fitness* 0,357828. Pada pengujian kombinasi *crossover rate (cr)* dan *mutation rate (cr)* menghasilkan solusi terbaik ketika nilai cr 0,4 dan nilai mr 0,6 dengan rata-rata nilai *fitness* 0,375874. Sedangkan pada pengujian perbandingan hasil antara GKA dengan GA tanpa *K-Means* yang dilakukan dengan menggunakan parameter yang sama, yaitu ukuran populasi 1600, jumlah generasi 800, nilai *cr* 0,4 dan *mr* 0,6, jumlah *sales* 3, serta pembagian titik kunjungan pada setiap *sales* adalah 92, 94, 89 memiliki hasil yang berbeda, di mana GKA menghasilkan total rute ketiga *sales* adalah sejauh 259,722337 km, sedangkan GA menghasilkan 1344,354477 km. Pada pengujian perbandingan GKA dengan rute yang digunakan perusahaan saat ini, GKA menggunakan hasil terbaiknya dengan total rute ketiga *sales* adalah sejauh 259,722337 km. sedangkan rute yang digunakan perusahaan saat ini memiliki total jarak tempuh 1113,520095 km.

7.2 Saran

Saran yang dapat digunakan untuk penelitian selanjutnya adalah sebagai berikut:

1. Parameter yang digunakan dalam sistem bisa dibuat lebih fleksibel lagi, seperti kondisi jalan yang macet yang akan dilewati, sehingga waktu yang dibutuhkan bisa menjadi lebih optimal. Hal tersebut bisa dilakukan dengan membuat *distance matriks* dengan *Google Maps* API yang berbayar.
2. Pada proses *crossover* ataupun mutasi bisa dicoba dengan membandingkan beberapa metode untuk mengetahui hasil terbaik, seperti Partial-Mapped Crossover (PMX), Cycle Crossocer (CX), dan lain sebagainya untuk *crossover*, serta Swap Mutation, Scrumble Mutation, Inverse Mutatioin, dan lain sebagainya untuk mutasi. Serta bisa juga dengan menambahkan teknik Adaptive Time Variant Genetic Algorithm seperti pada penelitian Seisarrina, M., Cholissodin, I., & Nurwarsito, H. (2018).

DAFTAR REFERENSI

A.J., U. and P.D., S., 2015. Crossover Operators in Genetic Algorithms: a Review. *ICTACT Journal on Soft Computing*, 06(01), pp.1083–1092.

Alfiyatin, A.N., Mahmudy, W.F. and Anggodo, Y.P., 2018. K-means clustering and genetic algorithm to solve vehicle routing problem with time windows problem. *Indonesian Journal of Electrical Engineering and Computer Science*, 11(2), pp.462–468.

Arikunto, S. and Jabar, A.S.C., 2018. *Evaluasi Program Pendidikan*.

Chopde, N. and Nichat, M., 2013. Landmark Based Shortest Path Detection by Using A* and Haversine Formula. *GH Raisoni College of Engineering and …*, [online] 1(2), pp.298–302. Available at: <http://www.ijircce.com/upload/2013/april/17_V1204030_Landmark_H.pdf>.

Karimah, S., Widodo, A.W. dan Cholissodin, I., 2017. Optimasi Multiple Travelling Salesman Problem Pada Pendistribusian Air Minum Menggunakan Algoritme Particle Swarm Optimization (Studi Kasus : UD . Tosa Malang). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, 1(9), pp.842–848.

Kotler, P., Wong, V., Saunders, J. dan Armstrong, G., 2005. *Principles of Marketing.* FOURTH EUR ed. *The Economic Journal*, Prentice Hall Europe.

Kumar, Y. and Sahoo, G., 2014. A New Initialization Method to Originate Initial Cluster Centers for K-Means Algorithm. 62, pp.43–54.

Larrañaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I. and Dizdarevic, S., 1999. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2), pp.129–170.

Mahmudy, W.F., 2016. Optimasi Multi Travelling Salesman Problem (M-TSP) Menggunakan Algoritma Genetika. *Seminar Nasional Basic Science*, [online] 1(2), pp.1–6. Available at: <http://wayanfm.lecture.ub.ac.id/files/2014/03/200802-Wayan-Basic-Science-mTsp-GAs.pdf>.

Mirjalili, S., Song Dong, J., Sadiq, A.S. dan Faris, H., 2020. *Genetic algorithm: Theory, literature review, and application in image reconstruction*. [online] *Studies in Computational Intelligence*, Springer International Publishing. Available at: <http://dx.doi.org/10.1007/978-3-030-12127-3_5>.

Muryadi, A.D., 2017. Model Evaluasi Program Dalam Penelitian Evaluasi. 3(1).

Pradnyana, G.A. and Permana, A.A.J., 2018. Sistem Pembagian Kelas Kuliah Mahasiswa Dengan Metode K-Means Dan K-Nearest Neighbors Untuk Meningkatkan Kualitas Pembelajaran. *JUTI: Jurnal Ilmiah Teknologi Informasi*, 16(1), p.59.

Rahman, A., Tan, H.I., Liew, W. and Shahruddin, N.S., 2014. Routing Mail Delivery from a Single Depot with Multiple Delivery Agents. 8(1), pp.15–34.

Rostami, A.S., Mohanna, F., Keshavarz, H. and Hosseinabadi, A.A.R., 2015. Solving multiple traveling salesman problem using the gravitational emulation local search algorithm. *Applied Mathematics and Information Sciences*, 9(2), pp.699–709.

Sarwono, Y.T., 2010. Aplikasi Model Jaringan Syaraf Tiruan Dengan Radial Basis Function Untuk Mendeteksi Kelainan Otak (Stroke Infark). *Jurnal Sistem Informasi*, pp.1–10.

Seisarrina, M.L., Cholissodin, I. and Nurwarsito, H., 2018. Invigilator Examination Scheduling using Partial Random Injection and Adaptive Time Variant Genetic Algorithm. *Journal of Information Technology and Computer Science*, 3(2), pp.113–119.

Vista, C.B. and Mahmudy, W.F., 2015. Penerapan algoritma evolution strategies untuk optimasi distribusi barang dua tahap. *Repository Jurnal Mahasiswa*, 5, no(11), p.12.

Zhanqing Lu, Kai Zhang, Juanjuan He, and Y.N., 2016. Applying K-means Clustering and Genetic Algorithm for Solving MTSP. 1, pp.271–277.

LAMPIRAN A DATA

Data set yang digunakan dalam penelitian ini berjumlah 275 Data.

No	Kode	Nama	Alamat	*latitude*	*longitude*
1	1008	A.YANI	A.YANI	-7.60747	111.8999
2	1011	ANEKA RASA	A.YANI 140138	-7.60808	111.8997
3	402124	KARMINI	WAGE PASAR - BARAT	-7.62022	111.8976
4	402400	SOFWAN BU	NGANJUK PASAR - BELAKANG	-7.60902	111.8986
5	402401	ABADI	NGANJUK PASAR - BELAKANG	-7.6098	111.8988
6	403557	SUNARTO TK.	KARTOHARJO	-7.60906	111.8989
7	493612	ZAENAL	PS WAGE UTARA	-7.6095	111.8992
8	493752	BU NYAMI	PASAR WAGE POJOK SELATAN	-7.61051	111.8986
9	493771	TK SUPARDI	PSR WAGE SELATAN	-7.61085	111.899
10	493785	PARIADI PAK	JLN. SERSAN HARUN KARTOHARJO	-7.60869	111.8955
11	493786	BAIHAQI	PAYAMAN	-7.60917	111.8988
12	493891	MAS ANTON	PASAR WAGE SELATAN - NGANJUK	-7.61073	111.8989
13	493956	SITI AMININ	PS. WAGE BELAKANG	-7.60911	111.8988
14	494347	ANA DIANITA PUSPITA SARI DEWI	Dk Bulu Rt 02 Rw 01 Babadan	-7.61042	111.8986
15	494381	BU RUBIYANTIE	Jln Sersan Harun No 70 A	-7.60902	111.8975
16	494498	TK. BINTI	PASAR WAGE	-7.60609	111.9006
17	493948	P. UDIN	PS. WAGE - NGANJUK	-7.60937	111.8992
18	402692	MAWAR BU BAMBANG	NGANJUK PASAR - BELAKANG	-7.60866	111.8985
19	494406	TOKO MASDA HIDAYAT	Jln Sersan Harun 27	-7.6089	111.8978
20	493700	TK HARIS	JL.BARITO NO 31 KERINGAN	-7.59896	111.9046
21	1026	PARLIN	WAGE PASAR	-7.6036	111.8937
22	402383	JADI	NGANJUK PASAR - BELAKANG	-7.60809	111.8994
23	402395	WARSINI DINA	NGANJUK PASAR - DALAM	-7.6092	111.8991
24	402396	YULI HIDAYATI	NGANJUK PASAR - DALAM	-7.6093	111.8992
25	403473	MARIATI	NGANJUK PSR	-7.60851	111.8995
26	403474	SUMIATI	NGANJUK PSR UTARA	-7.60853	111.8995
27	403554	SUBANDI	LETJEN SUPRAPTO 01 JL	-7.60842	111.8994

28	403555	YANI	S.PARMAN JL. IV RT 03 RW 05	-7.60847	111.8996
29	403563	TK.ENY	PERUMNAS CANDI REJO RT 04 RW05	-7.60863	111.8993
30	493606	IQBAL	DS MUKUNG RT 02 RW 06	-7.60909	111.8994
31	493607	DWI SRIYANI	PS WAGE UTARA	-7.60948	111.8993
32	493788	SUTARKO TK	PASAR WAGE UTARA	-7.60951	111.8993
33	493792	TK.SRI	DS CANGKRINGAN NGANJUK	-7.60945	111.8991
34	493884	ANDHITA	NGANJUK PASAR - DALAM	-7.60978	111.899
35	494273	CV. BOROBUDUR PRIMA SEJAHTERA I	JL YOS SUDARSO NO. 26	-7.60498	111.9008
36	494339	GEMIASIH	Stand Pasar Wage Nganjuk	-7.60849	111.8993
37	494362	SUSMIATI	Jln Letjend Suprapto No IA	-7.60217	111.9044
38	494427	TOKO MBAK DAM	Stand Pasar Wage Mboh sing di	-7.60867	111.8993
39	494529	NYOTO NIAGA JAYA	PASAR WAGE BARAT	-7.60871	111.8997
40	493739	TRI WAHYUNINGSIH	SUKOREJO RT 01 RW 03	-7.62691	111.9081
41	401384	CV. AHMAD YANI RAYA	A. YANI	-7.60603	111.9003
42	347	SETIA	A.YANI 96 TELP.91	-7.60536	111.9006
43	348	SUMBER AGUNG	A.YANI 102 TLP 195	-7.6054	111.9003
44	401034	CV. AHMAD YANI RAYA III	AIPDA S. TUBUN 5 NGANJUK	-7.59962	111.8947
45	402398	MAMIK BU	NGANJUK PASAR - DALAM	-7.60818	111.8996
46	402417	TARUNI	NGANJUK PASAR - DALAM	-7.60819	111.8996
47	493958	BU SUPARMI	JL MAYJEND SUNGKONO	-7.60921	111.8992
48	15	ANYAR	A.YANI 201 TLP 225	-7.61598	111.8862
49	494405	TOKO ABEL	Jln Kartini No 31 B	-7.60142	111.9036
50	415	BUDAYA	DIPONEGORO 46 TLP 21450	-7.60137	111.9046

Data Lengkap yang digunakan dalam penelitian ini dapat dilihat pada link berikut:

https://github.com/affand123/skripsi/blob/main/datasetfull.xlsx

LAMPIRAN B HASIL PENGUJIAN UKURAN POPULASI

Ukuran Populasi	*Fitness* Percobaan ke-										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
200	0,204786	0,202917	0,203	0,195404	0,21134	0,205262	0,195391	0,197152	0,197242	0,199793	0,201229
400	0,211157	0,202551	0,196434	0,197653	0,202489	0,208019	0,211319	0,198549	0,193142	0,207205	0,202852
600	0,217374	0,203787	0,199798	0,211467	0,217903	0,201608	0,213436	0,218677	0,210123	0,210428	0,21046
800	0,217167	0,210345	0,20736	0,217027	0,208247	0,213865	0,21598	0,20915	0,218015	0,205555	0,212271
1000	0,211382	0,217325	0,209558	0,221485	0,214376	0,201485	0,201808	0,208869	0,209446	0,209056	0,210479
1200	0,220762	0,221302	0,208154	0,213888	0,208696	0,216866	0,215934	0,212647	0,213213	0,214266	0,214573
1400	0,219044	0,21575	0,213545	0,207644	0,223396	0,219665	0,211534	0,207617	0,217293	0,215596	0,215108
1600	0,212504	0,210993	0,219184	0,215761	0,234635	0,228813	0,217672	0,216649	0,211407	0,210484	0,21781
1800	0,213919	0,236758	0,222363	0,200529	0,211587	0,20737	0,210593	0,222912	0,215528	0,220891	0,216245
2000	0,20447	0,202433	0,227072	0,206603	0,220131	0,212588	0,229772	0,224304	0,217955	0,232638	0,217797

LAMPIRAN C HASIL PENGUJIAN BANYAK GENERASI

Banyak Generasi	*Fitness* Percobaan ke-										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
200	0,297581	0,278075	0,275523	0,291222	0,282591	0,282267	0,285901	0,28528	0,285447	0,275996	0,283988
300	0,310883	0,335352	0,323795	0,309195	0,312724	0,326883	0,323253	0,30712	0,315223	0,312422	0,317685
400	0,341665	0,344867	0,350723	0,349482	0,36258	0,339561	0,324985	0,35284	0,364026	0,348084	0,347881
500	0,363987	0,34973	0,342058	0,351162	0,357263	0,349633	0,363398	0,339803	0,366722	0,341862	0,352562
600	0,311415	0,311894	0,315305	0,312908	0,317784	0,310268	0,322538	0,302929	0,317028	0,317967	0,314003
700	0,351719	0,350082	0,338127	0,358555	0,354837	0,366266	0,360285	0,368714	0,354126	0,358713	0,356142
800	0,354499	0,349074	0,363156	0,357922	0,375536	0,354042	0,353974	0,356429	0,368319	0,345329	0,357828
900	0,352246	0,348129	0,369242	0,356384	0,368911	0,360589	0,365023	0,346546	0,342948	0,345606	0,355562
1000	0,339057	0,342257	0,336115	0,35951	0,358584	0,358189	0,363327	0,361833	0,356803	0,36515	0,354083

LAMPIRAN D HASIL PENGUJIAN KOMBINASI *CROSSOVER RATE DAN MUTATION RATE*

cr : mr	*Fitness* Percobaan ke-										Rata-rata *Fitness*
	1	2	3	4	5	6	7	8	9	10	
0.1 ; 0.9	0,373691	0,367506	0,364649	0,366632	0,377255	0,360729	0,36547	0,359086	0,364338	0,332655	0,363201
0.2; 0.8	0,360465	0,359117	0,358174	0,365047	0,351498	0,368259	0,369421	0,378192	0,365761	0,374081	0,365002
0.3; 0.7	0,364291	0,367496	0,365584	0,374992	0,376082	0,365387	0,377869	0,377551	0,361806	0,355515	0,368657
0.4; 0.6	0,384198	0,366677	0,368464	0,36786	0,38362	0,385027	0,377361	0,372142	0,383839	0,369551	0,375874
0.5; 0.5	0,373712	0,379843	0,373815	0,361378	0,377876	0,368786	0,361409	0,36609	0,367249	0,364821	0,369498
0.6; 0.4	0,373691	0,380074	0,363962	0,373217	0,377506	0,375823	0,364587	0,36585	0,373187	0,360628	0,370853
0.7; 0.3	0,37884	0,378382	0,366438	0,376052	0,376906	0,38149	0,363888	0,363285	0,377212	0,379898	0,374239
0.8; 0.2	0,378695	0,375679	0,376522	0,346617	0,377546	0,360952	0,377368	0,361162	0,366315	0,362314	0,368317
0.9; 0.1	0,356016	0,381787	0,387604	0,380509	0,370473	0,365309	0,371241	0,375758	0,37523	0,367744	0,373167

LAMPIRAN E HASIL WAWANCARA

Wawancara dilakukan dengan salah satu perwakilan dari PT Indomarco Adi Prima (*Stock Point Nganjuk*).

DAFTAR PERTANYAAN WAWANCARA

Daftar pertanyaan wawancara ini berfungsi untuk menggali permasalahan pada penelitian yang berjudul **"Optimasi Multiple Travelling Salesmen Problem Distribusi Produk PT Indomarco Adi Prima (Stock Point Nganjuk) dengan menggunakan Metode K-Means dan Algoritma Genetika (GKA)"**. Berikut daftar pertanyaan wawancara dan jawaban pihak terkait.

Daftar pertanyaan dan jawaban:

Narasumber : Martino Andre Rosiyanto

Posisi/jabatan : Admin Sales

1. Apakah PT Indomarco Adi Prima Stock Point Nganjuk ini menangani distribusi produk?
   Jawaban: Ya betul, kami memang menangani distribusi produk di toko-toko atau minimarket.
2. Menangani distribusi produk apa PT Indomarco Adi Prima Stock Point Nganjuk?
   Jawaban: Produk yang kami distribusikan adalah produk kebutuhan sehari-hari terutama produk dari Indofood.
3. Apakah wilayah distribusi pada Stock Point ini hanya di wilayah Nganjuk?
   Jawaban: Ya benar, dari pusat memang sudah dibagi-bagi untuk wilayah distribusinya, di Kediri juga ada, di Jombang dan daerah lain juga ada.
4. Apakah dalam proses pendistribusian produk sudah ada pengaturan rute atau jalur yang akan dikunjungi dari PT Indomarco Adi Prima Stock Point Nganjuk?
   Jawaban: dari kami sudah ada daftar list rencana perjalanan sales untuk wilayah ini.
5. Apakah sudah ditentukan toko mana saja yang dikunjungi setiap harinya? atau dibebaskan yang penting dikunjungi semua?
   Jawaban: Dari kami sebenarnya sudah membuat list rencana perjalanan sales tersebut urut mulai dari list pertama hingga akhir, tetapi dalam hal ini kami membebaskan asal dikunjungi semua dalam seminggu hari kerja atau efektifnya 5 hari senin sampai jum'at.

6. Berapa jumlah kendaraan yang biasanya dipakai untuk distribusi produk tersebut?

   Jawaban: ada 3 wilayah yang kami handle, masing-masing kami tugaskan 1 kendaraan, jadi ada 3 kendaraan yang digunakan.

7. Ada berapa total titik atau jumlah toko yang dikunjungi dari ketiga kendaraan tersebut?

   Jawaban: masing-masing kendaraan biasanya ada sekitar 90-an titik yang akan dikunjungi dalam seminggu hari kerja atau efektifnya 5 hari itu.

8. Apakah para sales yang ditugaskan memperhatikan jarak tempuh distribusi?

   Jawaban: Kalau sales kami biasanya yang penting semua titik selesai dikunjungi.

9. Bagaimana pendapat anda sebagai admin sales apabila rute atau jalur distribusi yang dilalui para sales tersebut dioptimasi atau dicari kombinasi titik yang menghasilkan jarak yang terpendek?

   Jawaban: Saya sangat mendukung apabila jalur tersebut dapat dioptimasi, dampaknya juga ke sales yang bersangkutan, jadi akan lebih menghemat waktu dalam bekerja, dan untuk pihak perusahaan juga akan diuntungkan bila hal tersebut dapat terwujud.

Nganjuk, 5 November 2020
Admin Sales,

INDOMARCO
SP. NGANJUK

Martino Andre Rosiyanto